TOSHIBA

TMP68301

16ビット・マイクロプロセッサ・コアASSP
集積回路技術資料

目　次

16ビット・マイクロプロセッサ

TMP68301

第1章 はじめに

TMP68301は、68000のCMOS版である68HC000をコア・プロセッサとしてそのまま用い、周辺デバイスとしてシリアルI/F, パラレルI/F, タイマ, インタラプト・コントローラとアドレス・デコード回路等、標準的に68000の周辺に必要な回路を内蔵しています。

特徴としては、次の様なものがあります。

- 17個の32ビット・レジスタ
- 16Mバイトの直接アドレッシング
- 56種類の強力な基本命令
- 14種類のアドレッシング・モード
- シリアル I/F非同期通信 3ch
- パラレル I/F 16ビットのI/O(セントロニクス I/Fをサポート)
- タイマ 16ビットカウンタ 3ch
- インタラプト・コントローラ 10ch(外 3ch 内 7ch)
- CS信号
- ウェイトの自動挿入
- バス監視機能
- 低消費電力(CMOS)

TMP68301の構成を図1.1に示します。

TMP68301には、2つの動作モードがあります。一つは通常の動作をするノーマル・モード, もう一つは68000の開発ツール(ICE)を使うためのエミュレーション・モードです。エミュレーションモードでは68HC000コアは動かず、内蔵周辺デバイスは外から与えたアドレス/データ/制御信号によって動きます。

68HC000コアは、8ビット周辺デバイス制御用の信号である E, $\overline{\text{VPA}}$, $\overline{\text{VMA}}$が使えない他はTMP68HC000と同じですので、TMP68HC000を既に理解されている人は以下の周辺機能に関連した箇所を読んでください。

又、命令の詳細については、TMP68000と完全交換ですので、TLCS-68000ユーザーズマニュアルを参照してください。

第4章　信号およびバス動作

4.1　信号の説明

第6章　周辺機能ブロック

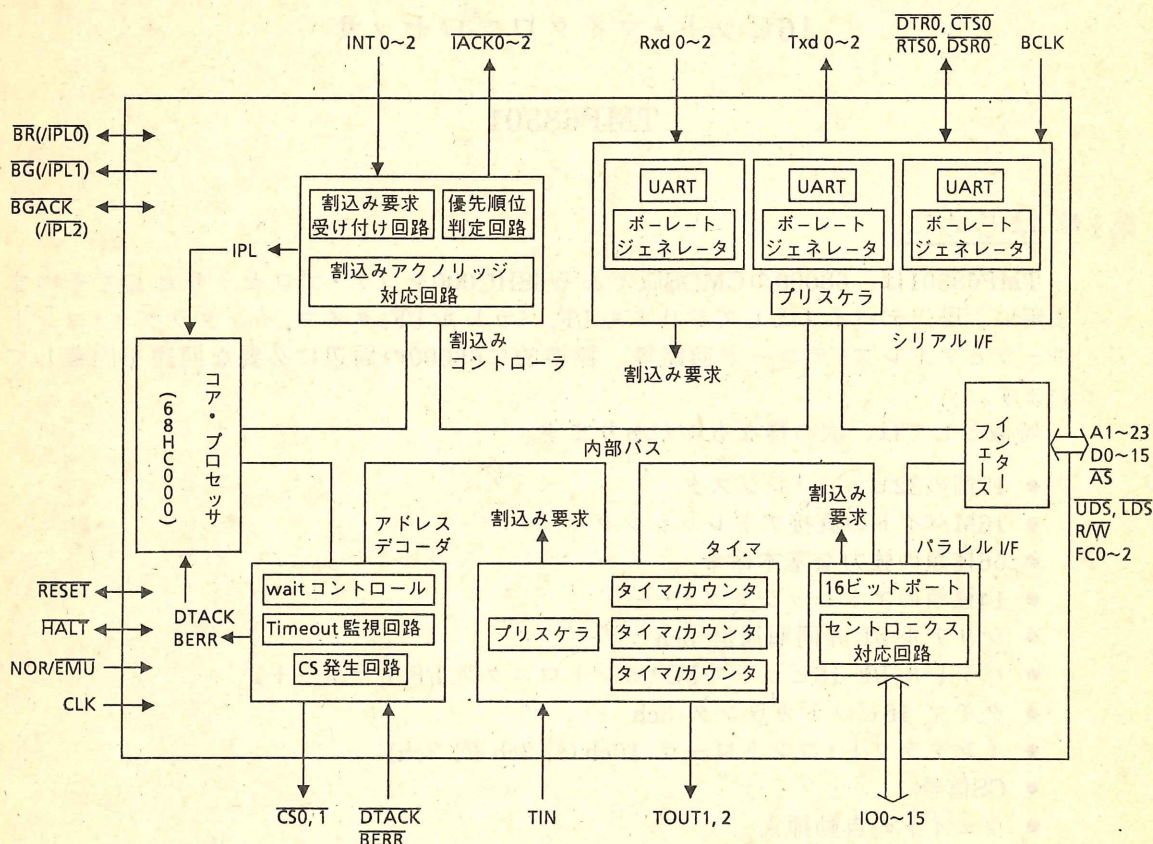



図1.1 TMP68301 ブロック図

プログラミング・モデル(図1.2, 1.3)に示すように、TMP68301には32ビットのレジスタが17個、32ビットのプログラム・カウンタ、および16ビットのステータス・レジスタ(下位バイトはコンディション・コード・レジスタ)があります。

図1.2 の最初の 8個のレジスタ (D0~D7) はデータ・レジスタで、バイト(8ビット)、ワード(16ビット)、およびロング・ワード(32ビット)のオペランドとして使われます。次の 7個のレジスタ(A0~A6)およびユーザ・スタック・ポインタ(USP) はソフトウェアのスタック・ポインタおよびベース・アドレス・レジスタとして使うことができます。それ以外に、このレジスタはワードおよびロング・ワードのオペランドとして使うことができます。16個のレジスタはすべてインデックス・レジスタとして使えます。(ユーザ・モード)

スーパバイザ・モードではステータス・レジスタの上位バイトおよびスーパバイザ・スタック・ポインタ(SSP) も自由に使用可能です。(図1.3)

ステータス・レジスタ(図1.4)には, エクステンド(X), 負符号(N), ゼロ(Z), オーバフロー(V) キャリー(C) の各フラグの他に割込みマスク・ビット(8レベル可能) が含まれています。この他、ステータス・ビットとしてトレース・モード(T) およびスーパバイザ・ステート(S) の各ビットがあります。

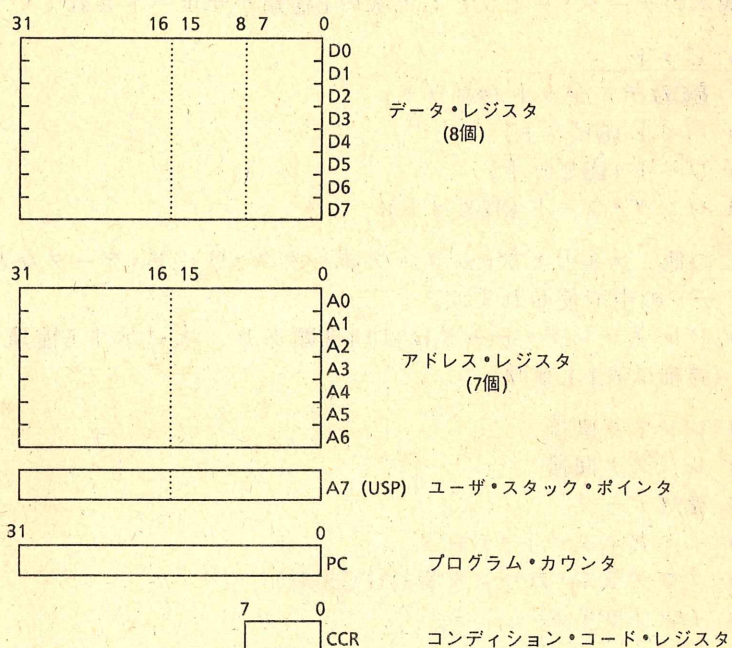



図1.2 ユーザ・プログラミング・モデル

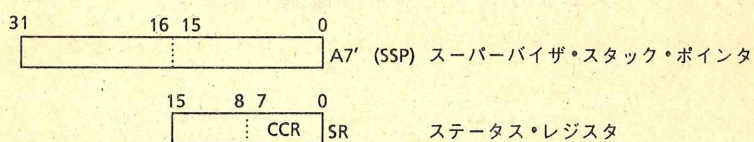



図1.3 スーパバイザ・プログラミング・モデル補足

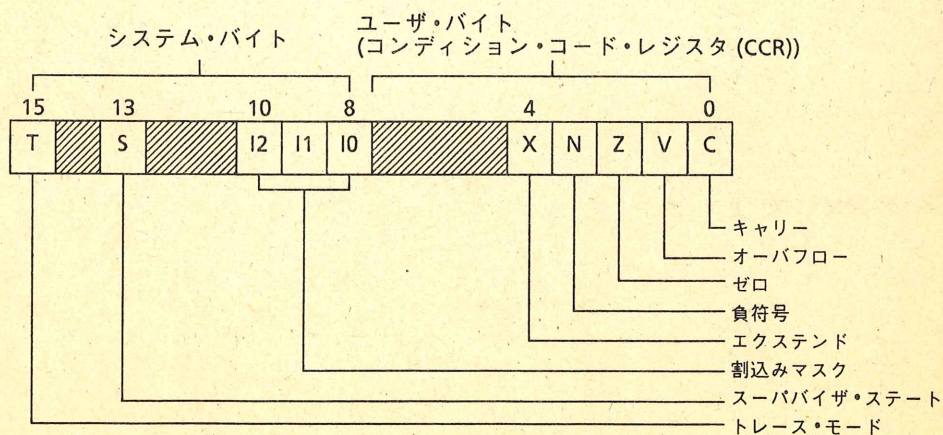



図1.4 ステータス・レジスタ

1.1 データ・タイプおよびアドレッシング・モード

基本のデータ・タイプとして次の5種類がサポートされています。

- ビット
- BCDディジット(4ビット)
- バイト(8ビット)
- ワード(16ビット)
- ロング・ワード(32ビット)

この他、メモリ・アドレス、ステータス・ワード・データなどのデータ・タイプも命令セットの中で使われます。

アドレッシング・モードには14種類あり、次に示す6種類が基本型になっています。(詳細は表1.1参照)

- レジスタ直接
- レジスタ間接
- 絶対データ
- イミディエート・データ
- プログラム・カウンタ相対(PC相対)
- インプライド

レジスタ間接アドレッシング・モードには、ポストインクリメント、プリデクリメント、オフセット、およびインデックスの各機能が含まれています。PC相対モードはインデックスおよびオフセットで修飾できます。

表1.1 アドレッシング・モード

アドレッシング・モード	表記法
レジスタ直接	
データ・レジスタ直接	Dn
アドレス・レジスタ直接	An
絶対データ	
絶対ショート	Abs.W
絶対ロング	Abs.L
PC相対	
オフセット付き	d16 (PC)
インデックスおよびオフセット付き	d8 (PC, Xn)
レジスタ間接	
レジスタ間接	(An)
ポストインクリメント・レジスタ間接	(An) +
プリデクリメント・レジスタ間接	- (An)
オフセット付きレジスタ間接	d16 (An)
インデックス＋オフセット付きレジスタ間接	d8 (An, Xn)
イミディエート・データ	
イミディエート	#xxx
クイック・イミディエート	#1～#8
インプライド	
インプライド・レジスタ	SR / USP / SSP / PC

(注)

Dn ＝データ・レジスタ
An ＝アドレス・レジスタ
Xn ＝アドレス・レジスタまたはデータ・レジスタ
SR ＝ステータス・レジスタ
PC ＝プログラム・カウンタ
SSP ＝スーパバイザ・スタック・ポインタ
USP ＝ユーザ・スタック・ポインタ
( ) ＝実効アドレス
d8 ＝8ビット・オフセット
d16 ＝16ビット・オフセット
#xxx ＝イミディエート・データ
Abs ＝絶対アドレス

1.2 命令セットの概要

TMP68301の命令セットを表1.2に示します。この他にこの中の命令の変型、またはサブセットがあり、それらは表1.3に示されています。命令セットはプログラム作成を容易にするため、構造化された高級言語をサポートし易いように特に考慮されています。各命令は、少数の例外を除いて、バイト、ワード、およびロング・ワードのオペランドを扱うことができ、ほとんどの命令で14種類のアドレッシング・モードがすべて使えます。命令のタイプ、データ・タイプ、およびアドレッシング・モードを組み合わせると、有効な命令は、1000種類以上にもなります。これらの命令の中には、符号付きおよび符号なしの乗除算、"クイック"算術演算、BCD演算、拡張処理(トラップによる)が含まれます。

表1.2 命令セット(1/2)

ニーモニック	内容
ABCD	エクステンド付き 10進加算
ADD	加算
AND	論理積
ASL	算術左シフト
ASR	算術右シフト
Bcc	条件付き分岐
BCHG	ビット・テストおよび変更
BCLR	ビット・テストおよびクリア
BRA	無条件分岐
BSET	ビット・テストおよびセット
BSR	サブルーチンへの分岐
BTST	ビット・テスト
CHK	レジスタの値の境界チェック
CLR	オペランドのクリア
CMP	比較
DBcc	条件のテスト, デクリメントおよび分岐
DIVS	符号付き除算
DIVU	符号なし除算
EOR	排他的論理和
EXG	レジスタ内容の交換
EXT	符号拡張
JMP	ジャンプ
JSR	サブルーチンへのジャンプ
LEA	実効アドレスのロード
LINK	スタックのリンク
LSL	論理的左シフト
LSR	論理的右シフト

表1.2 命令セット (2/2)

ニーモニック	内容
MOVE MOVEM MOVEP MULS MULU	転送 複数レジスタの転送 周辺データ転送 符号付き乗算 符号なし乗算
NBCD NEG NOP NOT	エクステンド付き10進数の補数化 補数化 ノーオペレーション 1の補数
OR	論理和
PEA	実効アドレスのプッシュ
RESET ROL ROR ROXL ROXR RTE RTR RTS	外部デバイスのリセット エクステンドなしの左ローテイト エクステンドなしの右ローテイト エクステンド付きの左ローテイト エクステンド付きの右ローテイト 例外処理からのリターン リターンおよびリストア サブルーチンからのリターン
SBCD Scc STOP SUB SWAP	エクステンド付き10進減算 条件付きセット ストップ 減算 データ・レジスタの上位ワードと下位ワードの交換
TAS TRAP TRAPV TST	テスト・アンド・セット トラップ オーバフローでのトラップ テスト
UNLK	スタックのアンリンク

表1.3　命令タイプの変型

命令タイプ	変　　型	内　　　容
ADD	ADD ADDA ADDQ ADDI ADDX	加算 アドレスの加算 クイック加算 イミディエート・データの加算 エクステンド付きの加算
AND	AND ANDI ANDI to CCR ANDI to SR	論理積 イミディエート・データとの論理積 イミディエート・データとCCRとの論理積 イミディエート・データとSRとの論理積
CMP	CMP CMPA CMPM CMPI	比較 アドレスの比較 メモリ内容の比較 イミディエート・データとの比較
EOR	EOR EORI EORI to CCR EORI to SR	排他的論理和 イミディエート・データとの排他的論理和 イミディエート・データとCCRとの排他的論理和 イミディエート・データとSRとの排他的論理和
MOVE	MOVE MOVEA MOVEQ MOVE from SR MOVE to SR MOVE USP	転送 アドレスの転送 クイック転送 ステータス・レジスタからの転送 ステータス・レジスタへの転送 ユーザ・スタック・ポインタの転送
NEG	NEG NEGX	補数化 エクステンド付きの補数化
OR	OR ORI ORI to CCR ORI to SR	論理和 イミディエート・データとの論理和 イミディエート・データとCCRとの論理和 イミディエート・データとSRとの論理和
SUB	SUB SUBA SUBI SUBQ SUBX	減算 アドレスの減算 イミディエート・データの減算 クイック減算 エクステンド付きの減算

第2章 データ構造およびアドレッシング・モード

この章では、TMP68301のレジスタおよびデータ構造について説明します。

2.1 オペランド・サイズ

オペランド・サイズは、バイト(8ビット)、ワード(16ビット)、およびロング・ワード(32ビット)があります。各命令のオペランドは命令の中で明示されている場合と、命令の動作によって暗黙に決められる場合があります。暗黙型の命令は、上記3種類のサイズのサブセットをサポートします。

2.2 レジスタ中のデータ構成

8個のデータ・レジスタによって1、8、16および32ビットのデータ・オペランドをサポートします。7個のアドレス・レジスタおよびスタック・ポインタは32ビットのアドレス・オペランドをサポートします。

2.2.1 データ・レジスタ

各データ・レジスタは32ビット長です。バイト・オペランドは下位8ビット、ワード・オペランドは下位16ビット、ロング・ワード・オペランドは32ビット全体を占めます。ビット0が最下位ビット、ビット31が最上位ビットです。

データ・レジスタがソースまたはデスティネーション・オペランドとして使われる時、該当するデータ・レジスタの下位部分だけが対象となります。その他の上位部分は使われず、変更されません。

2.2.2 アドレス・レジスタ

各アドレス・レジスタおよびスタック・ポインタは32ビット幅で、32ビットのフル・アドレスを保持します。アドレス・レジスタはバイト・サイズのオペランドをサポートしません。したがって、アドレス・レジスタがソース・オペランドとして使われる時、その下位ワードまたはロング・ワード全体が使われます。アドレス・レジスタがデスティネーション・オペランドとして使われる時は、オペランド・サイズとは無関係にレジスタ全体が影響を受けます。オペランド・サイズがワードの場合、他のオペランドは全て動作が実行される前に32ビットに符号拡張されます。

2.3 メモリ内のデータ構成

バイトは個々にアドレス可能で、ワードの中の上位バイトのアドレスが偶数(そのワードのアドレスと同じ)となります(図2.1)。下位バイトのアドレスは奇数で、そのワードのアドレスより1だけ大きい値です。命令および複数バイト・データはワード(偶数バイト)の境界でのみアクセス可能です。ロング・ワードのデータがアドレスn(nは偶数)にあれば、そのデータの第2ワードはアドレスn+2にあります。

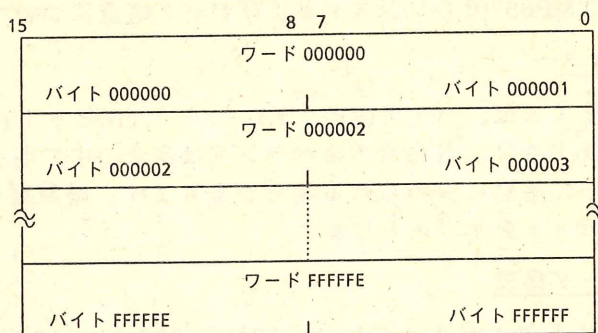



図2.1 メモリ内のワード構成

TMP68301はデータ・タイプとして、ビット・データ，8，16，または32ビットの整数データ，32ビットのアドレスおよびBCDデータをサポートします。各データ・タイプはメモリ内では図2.2のように置かれます。数字は、プロセッサからアクセスされる順序を示します。

図2.2 メモリ内のデータ構成

2.4 アドレッシング

TMP68301の命令には2種類の情報、すなわち、実行されるべき機能のタイプ、およびその機能の対象となるオペランドのロケーションが含まれています。ここでは、オペランドのロケーション(アドレス)を決める方法について説明します。

命令は次の2種類の方法のいずれかを使って、オペランドのロケーションを指定します。

- レジスタ指定 ― レジスタ番号を命令のレジスタ・フィールドで指示する。
- 実効アドレス ― 各種の実効アドレス・モードを使う。
- 暗黙の参照 ― 命令の中で特定のレジスタを使うことが暗黙のうちに指定されている。

2.5 命令のフォーマット

命令の長さは図2.3に示すように1ワードから5ワードまであります。命令の長さおよび実行されるべき動作は命令の最初のワード(オペレーション・ワードと呼ばれる)で決まります。残りのワードによって更にオペランドを指定します。これらのワードはイミディエート・オペランドまたは、オペレーション・ワードで指定された実効アドレス・モードに対する拡張部分です。

15	0
オペレーション・ワード	(動作およびモードの指定)
イミディエート・オペランド	(有る場合は 1~2 ワード)
ソースの実効アドレス部	(有る場合は 1~2 ワード)
デスティネーションのアドレス拡張部	(有る場合は 1~2 ワード)

図2.3 命令のフォーマット

2.6 プログラム/データ参照

TMP68301ではメモリ参照を2種類(プログラム参照とデータ参照)に区別しています。プログラム参照は、その名の通り、実行中のプログラムが入っているメモリの部分に対する参照です。データ参照はデータが入っているメモリの部分への参照です。一般に、オペランドはPC相対アドレッシング・モードを除き、データ空間から読み出されます。オペランドの書込みはすべてデータ空間に対して行われます。

2.7 レジスタの指定

命令の中のレジスタ・フィールドで、使われるレジスタが指定されます。他のフィールドは選択されたレジスタがアドレス・レジスタであるかデータ・レジスタであるか、およびそのレジスタの使われ方を指定します。

2.8 実効アドレス

大部分の命令はオペレーション・ワード中の実効アドレス・フィールドを使って1つのオペランドのロケーションを指定します。たとえば、単一の実効アドレスの命令のオペレーション・ワードの一般的フォーマットを図2.4に示します。

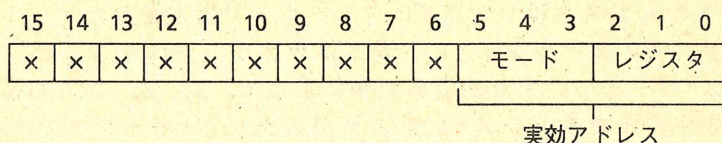



図2.4 単一実効アドレスの命令のオペレーション・ワードの一般形

実効アドレスは2つの3ビット・フィールド(モード・フィールドおよびレジスタ・フィールド)から構成されます。モード・フィールドの値はアドレス・モードの種類を示し、レジスタ・フィールドにはレジスタの番号が入っています。

オペランドを指定するのに、実効アドレス・フィールドの他に別の情報が必要になる場合もあります。この情報(実効アドレス拡張部)は次のワード(複数の場合もある)に含まれていて、命令の一部と考えられます(図2.3参照)。実効アドレス・モードは3種類(レジスタ直接、メモリ・アドレッシング、特殊)に分類されます。

2.8.1 レジスタ直接モード

このモードは、16個の汎用レジスタの1つがオペランドに指定されます。

2.8.1.1 データ・レジスタ直接

実効アドレスのレジスタ・フィールドで指定されたデータ・レジスタがオペランドとなります。

2.8.1.2 アドレス・レジスタ直接

実効アドレスのレジスタ・フィールドで指定されたアドレス・レジスタがオペランドとなります。

2.8.2 メモリ・アドレス・モード

このモードではメモリ内のオペランドのアドレスが指定されます。

2.8.2.1 アドレス・レジスタ間接

オペランドのアドレスは、レジスタ・フィールドで指定されたアドレス・レジスタの中にあります。この参照はJMP命令およびJSR命令を除きデータ参照に分類されます。

2.8.2.2 ポストインクリメント付きアドレス・レジスタ間接

オペランドのアドレスはレジスタ・フィールドで指定されたアドレス・レジスタの中にあります。そのオペランド・レジスタは使われた後、オペランドのサイズがバイト、ワード、またはロング・ワードのいずれであるかによって、1, 2, または4だけ増加されます。またアドレス・レジスタがスタック・ポインタでオペランドのサイズがバイトの場合、アドレスは2だけ(1ではなく)増加されます(スタック・ポインタをワードの境界に保つため)。この参照はデータ参照に分類されます。

2.8.2.3 プリデクリメント付きアドレス・レジスタ間接

オペランドのレジスタはレジスタ・フィールドで指定されたアドレス・レジスタの中にあります。そのオペランド・アドレスはオペランドのサイズがバイト、ワード、またはロング・ワードのいずれかであるかによって、1, 2, または4だけ減らされてから使われます。またアドレス・レジスタがスタック・ポインタでオペランドのサイズがバイトの場合、アドレスは2だけ(1ではなく)減らされます(スタック・ポインタをワードの境界に保つため)。この参照はデータに分類されます。

2.8.2.4 ディスプレースメント付きアドレス・レジスタ間接

このアドレス・モードでは拡張ワードが1個必要です。オペランドのアドレスは指定のアドレス・レジスタの内容に拡張ワード中の16ビット・ディスプレースメント(符号拡張されたもの)およびインデックス・レジスタの内容を加算したものとなります。この参照はJMP命令およびJSR命令を除きデータ参照に分類されます。

2.8.2.5 インデックス付きアドレス・レジスタ間接

このアドレス・モードでは拡張ワードが1個必要です。オペランドのアドレスは指定のアドレス・レジスタの内容に拡張ワード中の下位8ビット(符号拡張されたもの)およびインデックス・レジスタの内容を加算したものとなります。この参照はJMP命令およびJSR命令を除きデータ参照に分類されます。

2.8.3 特殊アドレス・モード

このモードでは実効アドレス・フィールドのレジスタ・フィールドを使って特殊なアドレッシング・モード(レジスタ番号ではなく)を指定します。

2.8.3.1 絶対ショート・アドレス

このアドレス・モードでは拡張ワードが1個必要です。その拡張ワード(符号拡張されて使われる)がオペランドのアドレスです。この参照はJMP命令およびJSR命令を除きデータ参照に分類されます。

2.8.3.2 絶対ロング・アドレス

このアドレス・モードでは拡張ワードが2個必要です。オペランドのアドレスは拡張ワードを連結することによって作られます。アドレスの上位16ビットが最初の拡張ワードで、下位16ビットが2番目の拡張ワードです。この参照はJMP命令およびJSR命令を除きデータ参照に分類されます。

2.8.3.3 ディスプレースメント付きPC

このアドレス・モードでは拡張ワードが1個必要です。オペランドのアドレスはPC値に拡張ワード中の16ビット・ディスプレースメント(符号拡張されたもの)を加算した値となります。PC値は拡張ワードのアドレスです。この参照はプログラム参照に分類されます。

2.8.3.4 インデックス付きPC

このアドレス・モードでは拡張ワードが1個必要です。アドレスはPC値，拡張ワードの下位8ビット(符号拡張されたもの)およびインデックス・レジスタの内容を加算した値となります。PC値は拡張ワードのアドレスです。この参照はプログラム参照に分類されます。

2.8.3.5 イミディエート・データ

このアドレス・モードではオペランドのサイズによって拡張ワードが1個または2個必要です。

バイトの場合	:	オペランドは拡張ワードの下位バイト。
ワードの場合	:	オペランドは拡張ワードそのもの。
ロング・ワードの場合	:	オペランドは2個の拡張ワード(上位16ビットが最初の拡張ワード、下位16ビットが2番目の拡張ワード)。

2.8.3.6 暗黙の参照

プログラム・カウンタ(PC)、スーパバイザ・スタック・ポインタ(SSP)、ユーザ・スタック・ポインタ(USP)、またはステータス・レジスタ(SR)を暗黙のうちに指定する命令があります。次の命令は実効アドレス・フィールドでSRを参照することができます。

CCR に対する ANDI
SR に対する ANDI
CCR に対する EORI
SR に対する EORI
CCR に対する ORI
SR に対する ORI
CCR への MOVE
SR への MOVE
SR からの MOVE

2.9 実効アドレスのエンコード要約

表2.1に前節の実効アドレッシング・モードの要約を示します。

表2.1 実効アドレスのエンコード要約

アドレッシング・モード	モード	レジスタ
データ・レジスタ直接	000	レジスタ番号
アドレス・レジスタ直接	001	レジスタ番号
アドレス・レジスタ間接	010	レジスタ番号
アドレス・レジスタ間接 (ポストインクリメント付き)	011	レジスタ番号
アドレス・レジスタ間接 (プリデクリメント付き)	100	レジスタ番号
アドレス・レジスタ間接 (ディスプレースメント付き)	101	レジスタ番号
アドレス・レジスタ間接 (インデックス付き)	110	レジスタ番号
絶対ショート	111	000
絶対ロング	111	001
ディスプレースメント付きPC	111	010
インデックス付きPC	111	011
イミディエート	111	100

2.10 システム・スタック

システム・スタックは多くの命令で暗黙に使われます。ユーザ・スタックおよびキューはアドレッシング・モードを使って生成され操作されます。アドレス・レジスタ(A7)がスタック・ポインタ(SP)です。SPはステータス・レジスタのSビットの状態によってスーパバイザ・スタック・ポインタ(SSP)またはユーザ・スタック・ポインタ(USP)のいずれかが使われます。Sビットがスーパバイザ・ステートを示している時はSSPがアクティブ・スタック・ポインタで、USPをアドレス・レジスタとして参照することはできません。 Sビットがユーザ・ステートを示している時はUSPがアクティブ・スタック・ポインタで，SSPは参照できません。システム・スタックはいずれもメモリ・アドレスの高い方から低い方へ向かって使われます。

第3章 命令セットの要約

この章ではTMP68301の命令セットの形式および構造の概要を説明します。命令セットは次の機能をすべて含みます。

データ転送
整数の算術演算
論理演算
シフトおよびローテイト
ビット操作
BCD演算
プログラム制御
システム制御

一連の命令群に前述の柔軟なアドレッシング・モードを結合することにより、プログラム開発に柔軟性を与えます。

3.1 データ転送の動作

データの転送および格納の基本的機能はMOVE命令によって実行されます。MOVE命令および実効アドレッシング・モードによってアドレスおよびデータの両方を操作することができます。データ転送命令により、バイト、ワード、およびロング・ワードのオペランドをメモリとメモリ、メモリとレジスタ、レジスタとレジスタの間で転送することができます。アドレス転送命令によってワードおよびロング・ワードのオペランドを転送することができ、正しいアドレス操作のみが実行されるようになっています。

一般的な転送命令の他に、特殊なデータ転送命令があります。それらは、MOVEM(複数レジスタの転送), MOVEP(周辺データ転送), EXG(レジスタ内容の交換), LEA(実効アドレスのロード), PEA(実効アドレスのプッシュ), LINK(スタックのリンク), UNLK(スタックのアンリンク)およびMOVEQ(クイック転送)です。表3.1にデータ転送動作の要約を示します。

表3.1 データ転送動作

命　　令	オペランド・サイズ	動　　　作
EXG	32	Xx↔Xy
LEA	32	EA→An
LINK	–	An→-(SP) SP→An SP+ディスプレースメント→SP
MOVE	8,16,32	s→d
MOVEM	16,32	(EA)→An,Dn An,Dn→(EA)
MOVEP	16,32	(EA)→Dn Dn→(EA)
MOVEQ	8	#xxx→Dn
PEA	32	EA→-(SP)
SWAP	32	Dn[31:16]↔Dn[15:0]
UNLK	–	An→SP (SP)+→An

(注) s : ソース　　　　　　　　-( ) : プリデクリメント付き間接
d : デスティネーション　　　( )+ : ポストインクリメント付き間接
[ ] : ビット番号　　　　　　#xxx : イミディエート・データ

3.2 整数の算術演算

算術演算には4種類の基本演算ADD(加算), SUB(減算), MUL(乗算), DIV(除算)の他にCMP(比較), CLR(クリア),およびNEG(補数)の各命令があります。加算および減算の命令はアドレスおよびデータの両方を扱うことができ、データの場合はすべてのオペランド・サイズが対象となります。アドレスの演算はアドレスとして許されるサイズ(16または32ビット)のオペランドに限定されます。データ、アドレス、およびメモリの比較演算もあります。クリアおよび補数化の命令はすべてのオペランド・サイズが使えます。乗算および除算は符号付きおよび符号なしのオペランドが使えます。乗算ではワードとワードを掛けてロング・ワードの積が得られ、除算ではロング・ワードの被除数をワードの除数で割って、ワードの商とワードの余りが得られます。

多倍精度および混合サイズの算術演算はエクステンド付きの命令を使って行うことができます。これらの命令にはADDX, SUBX, EXT, およびNEGXがあります。

TST(オペランドのテスト)命令(オペランドを0と比較した結果でコンディション・コードを設定する)もあります。TAS(テスト・アンド・セット)命令はマルチプロセッサ・システムで使われる同期用の命令です。表3.2に整数の算術演算の要約を示します。

表3.2 整数の算術演算

命　令	オペランド・サイズ	動　作
ADD	8,16,32 16,32	Dn + (EA) →Dn (EA) + Dn →(EA) (EA) + #xxx →(EA) An + (EA) →An
ADDX	8,16,32 16,32	Dx + Dy + X →Dx -(Ax) + -(Ay) + X →(Ax)
CLR	8,16,32	0 →(EA)
CMP	8,16,32 16,32	Dn-(EA) (EA) - #xxx (Ax) + -(Ay) + An-(EA)
DIVS	32 ÷ 16	Dn ÷ (EA)→Dn
DIVU	32 ÷ 16	Dn ÷ (EA)→Dn
EXT	8→16 16→32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16 × 16→32	Dn x (EA) →Dn
MULU	16 × 16→32	Dn x (EA) →Dn
NEG	8,16,32	0-(EA)→(EA)
NEGX	8,16,32	0-(EA)-X→(EA)
SUB	8,16,32 16,32	Dn-(EA) →Dn (EA)-Dn →(EA) (EA)-#xxx →(EA) An-(EA) →An
SUBX	8,16,32	Dx-Dy-X →Dx -(Ax) - -(Ay) - X →(Ax)
TAS	8	(EA)-0,1→EA[7]
TST	8,16,32	(EA)-0

[ ]　:　ビット番号
-( )　:　プリデクリメント付き間接
( )+　:　ポストインクリメント付き間接
#xxx　:　イミディエート・データ

3.3 論理演算

論理演算命令(AND, OR, EOR, およびNOT)はすべての整数データ・オペランドに対して適用されます。同様にイミディエート・データとの論理演算命令(ANDI, ORI, およびEORI)はすべてのサイズのイミディエート・データによる論理演算を実行することができます。論理演算の動作の要約を表3.3に示します。

表3.3 論理演算

命令	オペランド・サイズ	動作
AND	8, 16, 32	Dn∧(EA)→Dn (EA)∧Dn→(EA) (EA)∧#xxx→(EA)
OR	8, 16, 32	Dn∨(EA)→Dn (EA)∨Dn→(EA) (EA)∨#xxx→(EA)
EOR	8, 16, 32	(EA)⊕Dn→(EA) (EA)⊕#xxx→(EA)
NOT	8, 16, 32	~(EA)→(EA)

(注) ~ : 反転　　∨ : 論理和
#xxx : イミディエート・データ　　⊕ : 排他的論理和
∧ : 論理積

3.4 シフトおよびローテイト

算術シフト命令(ASR, ASL)および論理シフト命令(LSR, LSL)によって両方向へのシフト動作を行うことができます。ローテイト命令(エクステンド付きまたはエクステンドなし)はROXR, ROXL, ROR, およびROLが利用できます。シフト命令およびローテイト命令はレジスタまたはメモリのいずれにおいても実行できます。レジスタのシフトおよびローテイトではすべてのサイズのオペランドに適用され、シフト数は命令の中で指定される1~8ビット、またはデータ・レジスタで指定される0~63ビットが利用できます。

メモリのシフトおよびローテイトはワードオペランドだけで、1ビットだけシフト/ローテイトすることができます。表3.4にシフトおよびローテイトの動作の要約を示します。

表3.4 シフトおよびローテイトの動作

命　令	オペランド・サイズ	動　　作
ASL	8, 16, 32	X/C ← [←] ← 0
ASR	8, 16, 32	[→] → X/C
LSL	8, 16, 32	X/C ← [←] ← 0
LSR	8, 16, 32	0 → [→] → X/C
ROL	8, 16, 32	C ← [←]
ROR	8, 16, 32	[→] → C
ROXL	8, 16, 32	C ← [←] ← X
ROXR	8, 16, 32	X → [→] → C

3.5 ビット操作

ビット操作はBTST(ビット・テスト), BSET(ビット・テストおよびセット), BCLR(ビット・テストおよびクリア), BCHG(ビット・テストおよび変更)を使って実行することができます。表3.5にビット操作命令の動作の要約を示します。(ステータス・レジスタのビット2がZです。)

表3.5 ビット操作

命　令	オペランド・サイズ	動　　作
BTST	8, 32	~bit of (EA) →Z
BSET	8, 32	~bit of (EA) →Z 1 →bit of EA
BCLR	8, 32	~bit of (EA) →Z 0 →bit of EA
BCHG	8, 32	~bit of (EA) →Z ~bit of (EA) →bit of EA

3.6 BCD演算

複数桁のBCD数値についての算術演算はABCD(エクステンド付き10進加算), SBCD(エクステンド付き10進減算), およびNBCD(エクステンド付き10進補数化)を使って行うことができます。表3.6にBCD演算動作の要約を示します。

表3.6 BCD演算

命　令	オペランド・サイズ	動　　作
ABCD	8	$Dx_{10} + Dy_{10} + X \rightarrow Dx$ $-(Ax)_{10} + -(Ay)_{10} + X \rightarrow (Ax)$
SBCD	8	$Dx_{10} - Dy_{10} - X \rightarrow Dx$ $-(Ax)_{10} - -(Ay)_{10} - X \rightarrow (Ax)$
NBCD	8	$0-(EA)_{10}-X \rightarrow (EA)$

(注)
-(　)：　プリデクリメント・アドレッシング・モード

3.7　プログラム制御動作

プログラム制御動作は条件付きおよび無条件分岐命令およびリターン命令を使って実行することができます。これらの命令の要約を表3.7に示します。

条件命令は次の条件によって分岐します。

CC	キャリー・クリア	LS	LOWまたは等しい
CS	キャリー・セット	LT	より小さい
EQ	等しい	MI	マイナス
F	真でない	NE	等しくない
GE	大きいか等しい	PL	プラス
GT	より大きい	T	常に真
HI	HIGH	VC	オーバフローなし
LE	小さいか等しい	VS	オーバフロー

表3.7 プログラム制御動作

命　令	動　　作
条件付き Bcc	条件付き分岐(14条件) 8および16ビットのディスプレースメント
DBcc	条件をテストし,デクリメントして分岐(16条件) 16ビットのディスプレースメント
Scc	条件によってバイトを設定する(16条件)
無条件 BRA	常に分岐, 8および16ビットのディスプレースメント
BSR	サブルーチンへの分岐 8および16ビットのディスプレースメント
JMP	ジャンプ
JSR	サブルーチンへのジャンプ
リターン RTR	リターンおよびコンディション・コードの復帰
RTS	サブルーチンからのリターン

3.8 システム制御動作

システム制御動作は特権命令、トラップ発生命令、ステータス・レジスタを使用/変更する命令によって実行されます。これらの命令の要約を表3.8に示します。

表3.8 システム制御動作

命令	動作
特権命令	
RESET	外部デバイスをリセットする
RTE	例外処理からのリターン
STOP	プログラムの実行をストップする
ANDI to SR	ステータス・レジスタに対するAND演算
EORI to SR	ステータス・レジスタに対するEOR演算
ORI to SR	ステータス・レジスタに対するOR演算
MOVE USP	ユーザ・スタック・ポインタの転送
MOVE EA to SR	新しいステータスをSRにロードする
トラップ発生	
TRAP	トラップ
TRAPV	オーバフロー時トラップ
CHK	レジスタ内容を境界値に対してチェックする
ステータス・レジスタ	
ANDI to CCR	コンディション・コードに対するAND演算
EORI to CCR	コンディション・コードに対するEOR演算
ORI to CCR	コンディション・コードに対するOR演算
MOVE EA to CCR	コンディション・コードに新しい値をロード
MOVE SR to EA	ステータス・レジスタの格納

第4章 信号およびバス動作

この章では入力および出力信号について簡単に説明します。また各種のマシン・サイクルにおけるバス動作についても説明します。

(注)

以下「アサート(アサーション)」および「ネゲート(ネゲーション)」という用語をよく使います。これは"HIGHアクティブ", "LOWアクティブ"信号などが入り乱れて使われる時に混乱を避けるのが目的です。「アサート」または「アサーション」は電圧のLOWまたはHIGHに関係なく、信号がアクティブまたは真であることを示すために使われます。「ネゲート」または「ネゲーション」は信号が非アクティブまたは偽であることを示すために使われます。

4.1 信号の説明

ピン配置を図4.1に示します。

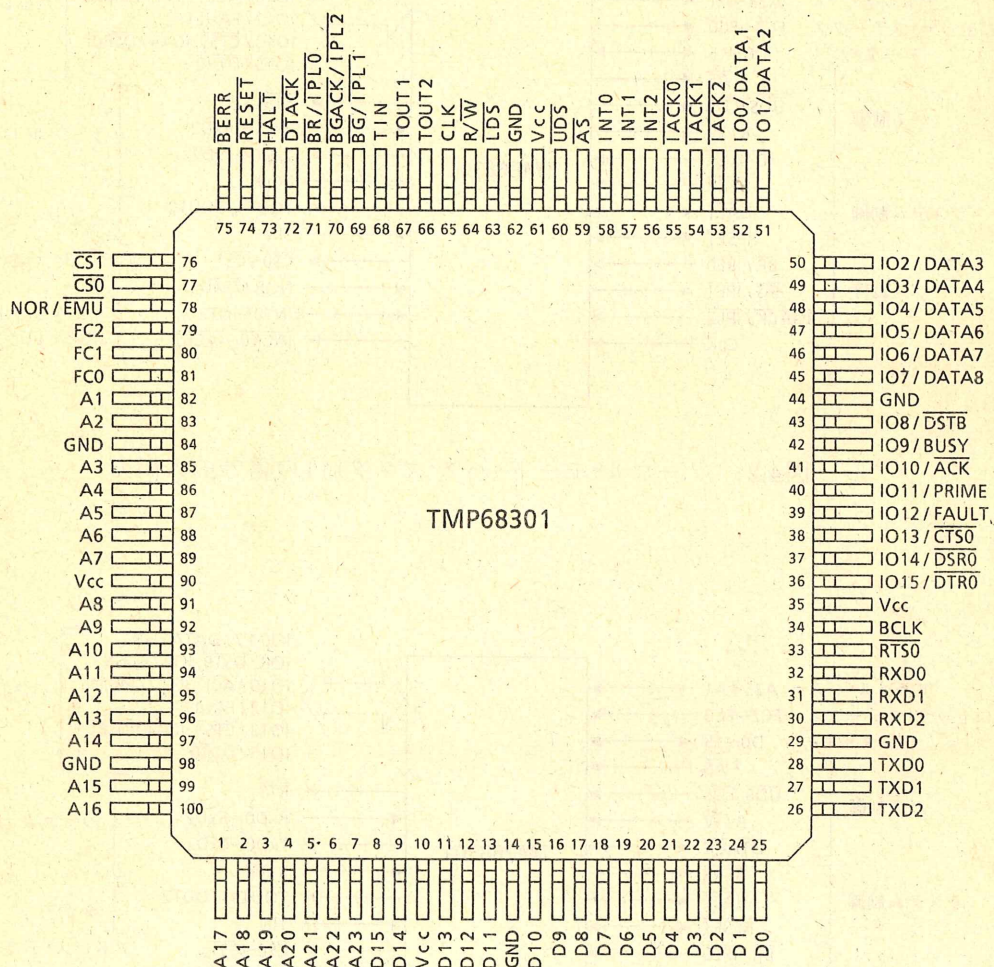



図4.1　PIN配置図 (TOP VIEW)

次に信号について簡単に説明します。

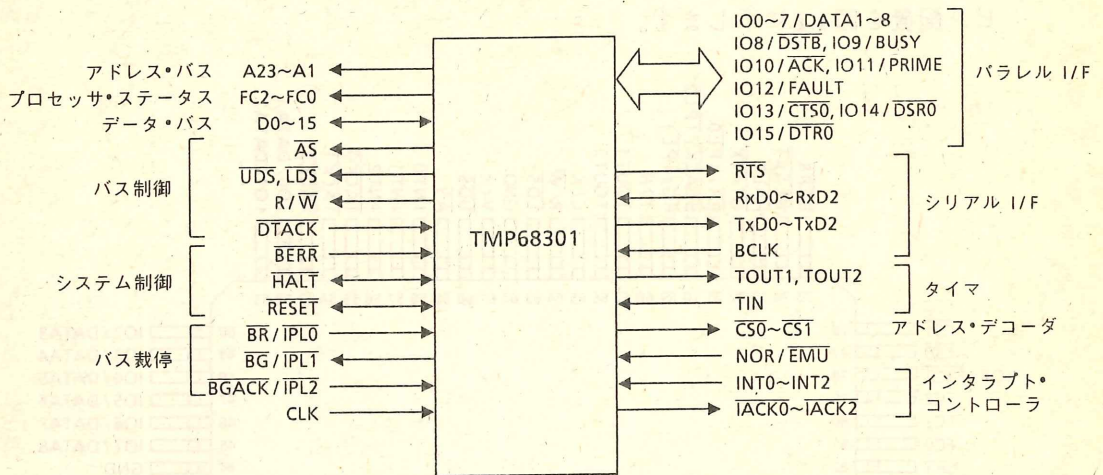



図4.2　ノーマル・モード(バス・マスタ時)の信号の入出力

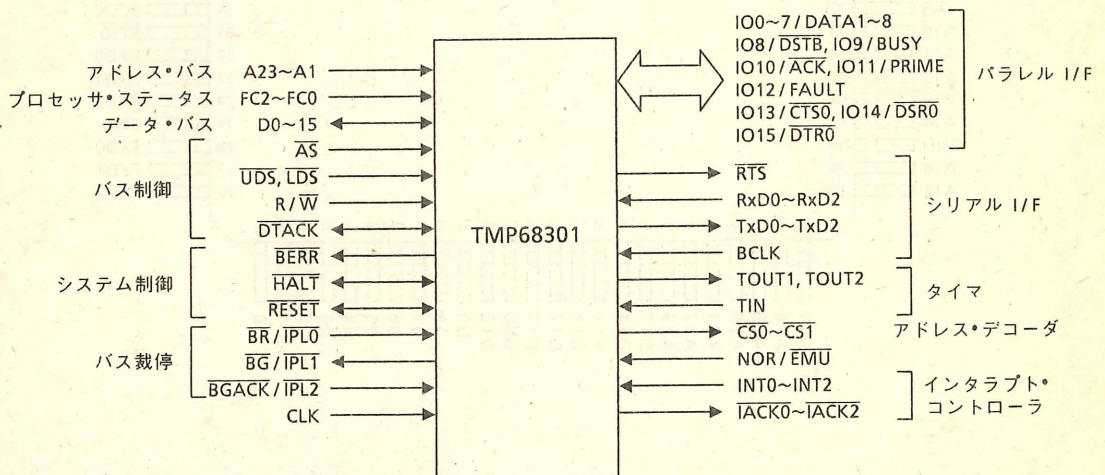



図4.3　ノーマル・モード(バス解放時)の信号の入出力

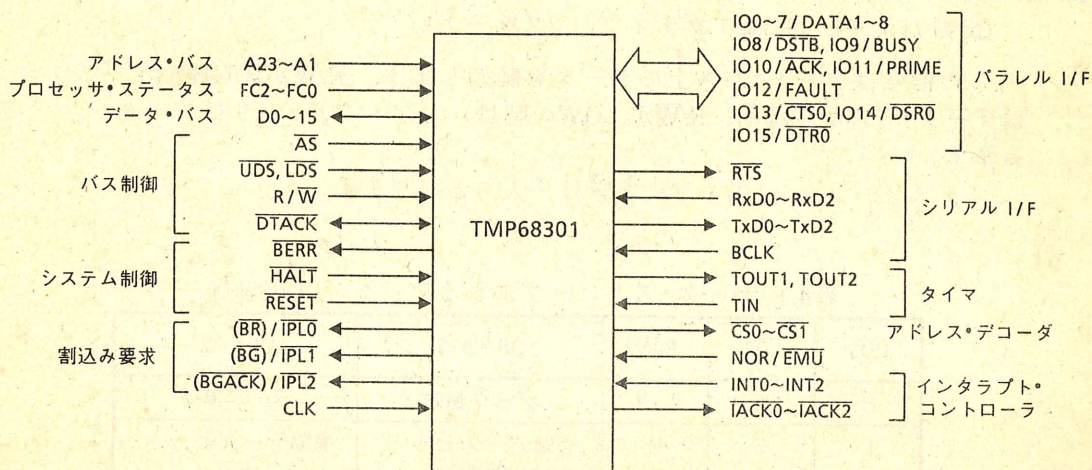



図4.4　エミュレーション・モードの信号の入出力

4.1.1 A1~A23 :「アドレス・バス」(出力/[入力], 3ステート)

この23ビットのバスは8メガ・ワードのデータをアドレッシングできます。

割込みアクノリッジ・サイクル以外のすべてのサイクルで、バス動作のためのアドレスを出力します。割込みアクノリッジ・サイクルサイクルではどのレベルの割込みがサービスされているかがA1~A3で示され、A4~A23はすべてHIGHとなります。

バスを解放したときおよびエミュレーション・モードのときは、入力に変わります。

4.1.2 D0~D15 :「データ・バス」(入出力, 3ステート)

この16ビットのバスは汎用のデータ・バスです。

ワード単位またはバイト単位でデータを転送することができます。割込みアクノリッジ・サイクルでは、外部デバイスがベクタ番号をD0~D7に乗せます。

エミュレーション・モードのときは、データ転送の方向が逆になります。

4.1.3 非同期のバス制御信号

4.1.3.1 $\overline{AS}$:「アドレス・ストローブ」(出力/[入力], LOWアクティブ, 3ステート)

この信号はアドレス・バス上に有効なアドレスが乗っていることを示します。

エミュレーション・モードのときは、入力となります。

4.1.3.2 R/$\overline{W}$:「読出し/書込み」(出力/[入力], 3ステート)

この信号はバス転送が読出し(HIGH)か書込み(LOW)かを示します。この信号は表4.1に示すように$\overline{UDS}$および$\overline{LDS}$と組み合わせて使われます。

エミュレーション・モードのときは、入力となります。

4.1.3.3 $\overline{\text{UDS}}$, $\overline{\text{LDS}}$:「上位および下位データ·ストローブ」(出力/[入力], LOWアクティブ, 3ステート)

この信号はデータ・バス上のデータを制御します。R/$\overline{\text{W}}$がHIGHの時、プロセッサはデータ·バスから読み、R/$\overline{\text{W}}$がLOWの時は、コア・プロセッサはデータ·バスへ書込みます。

エミュレーション・モードのときは、入力となります。

表4.1 データ・ストローブによるデータ·バスの制御

$\overline{\text{UDS}}$	$\overline{\text{LDS}}$	R/$\overline{\text{W}}$	D8~D15	D0~D7
H	H	-	データ無効	データ無効
L	L	H	有効データ·ビット 8~15	有効データ·ビット 0~7
H	L	H	データ無効	有効データ·ビット 0~7
L	H	H	有効データ·ビット 8~15	データ無効
L	L	L	有効データ·ビット 8~15	有効データ·ビット 0~7
H	L	L	有効データ·ビット 0~7*	有効データ·ビット 0~7
L	H	L	有効データ·ビット 8~15	有効データ·ビット 8~15*

L: LOW H: HIGH

*: この条件は現在のインプルメンテーションの結果であり、将来無くなる可能性があります。

4.1.3.4 $\overline{\text{DTACK}}$:「データ転送アクノリッジ」(入力/[出力], LOWアクティブ)

この信号はデータ転送が完了したことを示します。コア・プロセッサは読出しサイクルの間に$\overline{\text{DTACK}}$がアサートされたことを認めると、データをラッチし、バスサイクルを終了します。書込みサイクルの間に$\overline{\text{DTACK}}$が認められると、そのバス・サイクルは終了します。(4.4「非同期動作と同期動作」参照)。

エミュレーション・モードのときは、オープンドレインの出力となります。

4.1.4 バス裁停コントロール(割込み要求出力)

この3個の信号はどのデバイスがマスタになるかを決めるためのバス裁停回路で使われます。

また、エミュレーション・モードのときは、インタラプト・コントローラからの割込み要求出力となります。

4.1.4.1 $\overline{\mathrm{BR}}$ ($\overline{\mathrm{IPL0}}$) :「バス・リクエスト」(入力/[出力], LOWアクティブ)

この信号はバス・マスタになり得る他のすべてのデバイス信号とワイヤードORされます。この入力信号はどれか他のデバイスがバスの制御権を要求していることを示します。

エミュレーション・モードのときは、$\overline{\mathrm{IPL0}}$の出力となります。

4.1.4.2 $\overline{\mathrm{BG}}$ ($\overline{\mathrm{IPL1}}$) :「バス・グラント」(出力, LOWアクティブ)

この信号はバス・マスタになり得る他のすべてのデバイスに対して、コア・プロセッサがバスの制御を現在実行中のバス・サイクルの終りで解放することを示します。

エミュレーション・モードのときは、$\overline{\mathrm{IPL1}}$の出力となります。

4.1.4.3 $\overline{\mathrm{BGACK}}$ ($\overline{\mathrm{IPL2}}$):「バス・グラント・アクノリッジ」(入力/[出力], LOWアクティブ)

この信号はどれか他のデバイスがバス・マスタになったことを示します。この信号は次の4つの条件が満足されるまではLOWにすることはできません。

1. $\overline{\mathrm{BG}}$がアサートされている。
2. $\overline{\mathrm{AS}}$がアサートされていない(すなわち、コア・プロセッサがバスを使っていない)
3. $\overline{\mathrm{DTACK}}$がアサートされていない(すなわち、メモリまたは周辺デバイスがどれもバスを使っていない)
4. $\overline{\mathrm{BGACK}}$がアサートされていない(他のデバイスがバスの制御権を要求していない)

エミュレーションモードのときは、$\overline{\mathrm{IPL2}}$の出力となります。

4.1.5 システム・コントロール

4.1.5.1 $\overline{\mathrm{BERR}}$:「バス・エラー」(入力/[出力], LOWアクティブ)

この信号は現行のサイクルに問題があることをコア・プロセッサに知らせます。問題としては次の場合などがあります。

1. デバイスからの応答がない。
2. 割込みベクタ番号取込みのエラー
3. メモリ管理ユニットによって決定される不当なアクセス要求
4. 他のエラー(アプリケーションにより異なる)

$\overline{\mathrm{BERR}}$は$\overline{\mathrm{HALT}}$と関連し、例外処理を行うか現行バス・サイクルを再実行するかが決められます。(4.2.4「バス・エラーおよびホルト動作」参照)

エミュレーション・モードのときは、オープンドレインの出力となります。

4.1.5.2 $\overline{\text{RESET}}$:「リセット」(入出力, LOWアクティブ, オープン・ドレイン)

この信号は外部からのリセット信号の場合、コア・プロセッサをリセットする(システムの初期化シーケンスを開始する)ように働きます。内部で発生されるリセット(RESET命令による)の場合、外部デバイスに対するリセット信号として働き、コア・プロセッサの内部状態は変化しません。システム・リセット(コア・プロセッサおよび外部デバイス)は外部から $\overline{\text{HALT}}$ および $\overline{\text{RESET}}$ を同時にLOWにすることによって行われます。(4.2.5「リセット動作」参照)

4.1.5.3 $\overline{\text{HALT}}$:「ホルト」(入出力, LOWアクティブ, オープン・ドレイン)

この信号は外部デバイスからアサートされた場合、現行バス・サイクルの終了時にコア・プロセッサを停止させます。この入力信号を使ってプロセッサが停止させられた場合、コア・プロセッサからの制御信号はすべてネゲートされ、3ステート信号はすべてハイ・インピーダンス状態となります。($\overline{\text{BERR}}$ との関係については4.2.4「バス・エラーおよびホルト動作」参照。)

二重バス・エラー状態(4.2.4.4「二重バス・エラー」参照)が発生した場合などで、コア・プロセッサが命令の実行を停止すると $\overline{\text{HALT}}$ がコア・プロセッサによってLOWにドライブされ、コア・プロセッサが停止したことを外部デバイスにしらせます。

4.1.6 FC0, FC1, FC2:「機能コード」(出力/[入力], HIGHアクティブ, 3ステート)

これら3個の信号は表4.2に示すようにコア・プロセッサのステート(ユーザまたはスーパバイザ)および現行のサイクルの種類を示します。これらの信号で示される情報は $\overline{\text{AS}}$ がアサートされている時に有効です。

表4.2 機能コード出力

FC2	FC1	FC0	サイクルの種類
L	L	L	*
L	L	H	ユーザ・データ
L	H	L	ユーザ・プログラム
L	H	H	*
H	L	L	*
H	L	H	スーパバイザ・データ
H	H	L	スーパバイザ・プログラム
H	H	H	割込みアクノリッジ

L: LOW　H: HIGH
*: 未定義,将来使用予定

エミュレーション・モードのときは、入力となります。

4.1.7 CLK:「クロック」(入力, HIGHアクティブ)

この信号はTTLコンパチブルの信号で内部でバッファされてコア・プロセッサに必要な内部クロックが作られます。クロック入力は停止してはならず, さらにパルス幅の最小値および最大値の条件を満足していなければなりません。

4.1.8 周辺信号

4.1.8.1　IO0~IO7 / DATA1~DATA8「IOポート」(入出力)

汎用のIOポート，またはセントロニクスI/F用のデータ・バスです。

4.1.8.2　IO8 / $\overline{\mathrm{DSTB}}$「IOポート」(入出力)

汎用のIOポート，またはセントロニクスI/F用のストローブ信号です。

4.1.8.3　IO9 / BUSY「IOポート」(入出力)

汎用のIOポート，又はセントロニクスI/F用のビジー信号です。

4.1.8.4　IO10 / $\overline{\mathrm{ACK}}$「IOポート」(入出力)

汎用のIOポート，またはセントロニクスI/F用のアクノリッジ信号です。

4.1.8.5　IO11 / PRIME「IOポート」(入出力)

汎用のIOポート，またはセントロニクスI/F用のプライム信号です。

4.1.8.6　IO12 / FAULT「IOポート」(入出力)

汎用のIOポート，またはセントロニクスI/F用のフォルト信号です。

4.1.8.7　IO13 / $\overline{\mathrm{CTS0}}$「IOポート」(入出力)

汎用のIOポート，またはシリアルI/Fチャネル0のCTS信号です。

4.1.8.8　IO14 / $\overline{\mathrm{DSR0}}$「IOポート」(入出力)

汎用のIOポート，またはシリアルI/Fチャネル0のDSR信号です。

4.1.8.9　IO15 / $\overline{\mathrm{DTR0}}$「IOポート」(入出力)

汎用のIOポート，またはシリアルI/Fチャネル0のDTR信号です。

4.1.8.10　TOUT1, TOUT2「タイマ出力」(出力)

タイマのチャネル1, 2の出力信号です。

4.1.8.11　TIN「タイマ入力」(入力，LOWアクティブ)

タイマ各チャネルへの入力信号です。

4.1.8.12　$\overline{\mathrm{RTS0}}$「送信要求」(出力)

シリアルI/Fチャネル0のRTS信号です。

4.1.8.13　RxD0, RxD1, RxD2「受信データ」(入力)

シリアルI/Fのデータ入力です。

4.1.8.14　TxD0, TxD1, TxD2「送信データ」(出力)

シリアルI/Fのデータ出力です。

4.1.8.15　BCLK「ボーレート・クロック」(入力)

シリアルI/Fのボーレート発生用のクロックです。

4.1.8.16　INT0, INT1, INT2「割込み要求」(入力)

割込み要求です。

4.1.8.17　$\overline{\mathrm{IACK0}}$, $\overline{\mathrm{IACK1}}$, $\overline{\mathrm{IACK2}}$「割込みアクノリッジ」(出力，LOWアクティブ)

IACKサイクルであることを示します。

4.1.8.18　$\overline{\mathrm{CS0}}$, $\overline{\mathrm{CS1}}$「チップセレクト」(出力，LOWアクティブ)

アドレスをデコードした信号です。

4.1.9　NOR/$\overline{\mathrm{EMU}}$「モード切換」(入力)

ノーマル・モード,エミュレーション・モードの切り替え信号です。
ローでエミュレーション・モードになり、内部の68HC000コアはバスから切り離され、バス制御信号は図4.4のようになります。内蔵されている周辺回路は、外から入るアドレスや制御信号で動きます。

4.1.10　信号の要約

表4.3　信号の要約

信　号　名	ニーモニック	入力/出力	アクティブ状態	3ステート	ハイ・インピーダンス	
					$\overline{\text{HALT}}$	$\overline{\text{BG}}$
アドレス・バス	A1~A23	出(入)	H	Y	Y	Y
データ・バス	D0~D15	入出(出入)	H	Y	Y	Y
アドレス・ストローブ	$\overline{\text{AS}}$	出(入)	L	Y	N	Y
読出し/書込み	R/$\overline{\text{W}}$	出(入)	H(読出し) L(書込み)	Y	N	Y
上位および下位のデータ・ストローブ	$\overline{\text{UDS}}$, $\overline{\text{LDS}}$	出(入)	L	Y	N	Y
データ転送アクノリッジ	$\overline{\text{DTACK}}$	入(出1)	L	N1	N1	N1
バス・リクエスト	$\overline{\text{BR}}$ (/$\overline{\text{IPL0}}$)	入(出)	L	N	N	N
バス・グラント	$\overline{\text{BG}}$ (/$\overline{\text{IPL1}}$)	出	L	N	N	N
バス・グラント・アクノリッジ	$\overline{\text{BGACK}}$ (/$\overline{\text{IPL2}}$)	入(出)	L	N	N	N
バス・エラー	$\overline{\text{BERR}}$	入(出1)	L	N1	N1	N1
リセット	$\overline{\text{RESET}}$	入出	L	Y	N1	N1
停止	$\overline{\text{HALT}}$	入出	L	Y	N1	N1
機能コード	FC0, FC1, FC2	出(入)	H	Y	N	Y
クロック	CLK	入	H	N	N	N
IOポート	IO0~15	入出	—	—	—	—
タイマ出力	TOUT1, TOUT2	出	—	—	—	—
タイマ入力	TIN	入	L	—	—	—
送信要求	$\overline{\text{RTS0}}$	出	L	—	—	—
受信データ	RxD0~2	入	—	—	—	—
送信データ	TxD0~2	出	—	—	—	—
ボーレート・クロック	BCLK	入	—	—	—	—
モード切換え	NOR/$\overline{\text{EMU}}$	入	—	—	—	—
割込み要求	INT0~2	入	H	—	—	—
割込みアクノリッジ	$\overline{\text{IACK0}}$~2	出	L	—	—	—
チップ・セレクト	$\overline{\text{CS0}}$, $\overline{\text{CS1}}$	出	L	—	—	—
電源	Vcc	入	—	—	—	—
グランド	GND	入	—	—	—	—

(注)　1　オープンドレイン　　　　入力/出力の(　)内はエミュレーション・モードのとき

入　：入力
出　：出力　　H：HIGH　　Y：Yes
入出：入出力　L：LOW　　N：No

4.2 バス動作

4.2.1 データ転送動作

データ転送には次の信号が関係します。

1. アドレス・バス(A1~A23)
2. データ・バス(D0~D15)
3. 制御信号

アドレスとデータのバスは別々になっており、非同期のバス構造を使ってデータが転送されます。すべてのサイクルにおいて、バス・マスタはサイクルの始めと終りの両方で自分が出力するすべての信号のスキューを回避する責任を持つものと仮定します。この他、バス・マスタはスレーブ・デバイスからのアクノリッジおよびデータ信号のスキューを回避する責任があります。

次に読出し、書込み、およびリード・モディファイ・ライト(RMW)の各サイクルについて説明します。RMWサイクルでは読出し、変更、および書込みが連続して行われます。TMP68301はこのサイクルをインターロック型のマルチプロセッサ間通信に使います。

4.2.1.1 読出しサイクル

このサイクルでは、コア・プロセッサはメモリまたは周辺デバイスからデータを受け取ります。コア・プロセッサはすべての場合において複数バイトのデータを読みます。命令でワード(またはロング・ワード)の動作が指定されている時は、コア・プロセッサは両方(上位および下位)のバイトを読みます。バイト動作を指定している命令の場合は、コア・プロセッサは内部のA0ビットを使ってどちらのバイトを読むかを決定し、そのバイトに必要なデータ・ストローブを出力します。すなわち、A0=0の時は$\overline{\text{UDS}}$が, A0=1の時は$\overline{\text{LDS}}$が出力されます。読み出したバイトは命令で指定された場所へコア・プロセッサの内部で転送されます。

ワードの読出しサイクルのフローチャートを図4.5に、バイトの場合を図4.6に示します。読出しおよび書込みサイクルのタイミングを図4.7に示します。ワードおよびバイトの読出しサイクルのタイミングを図4.8に示します。

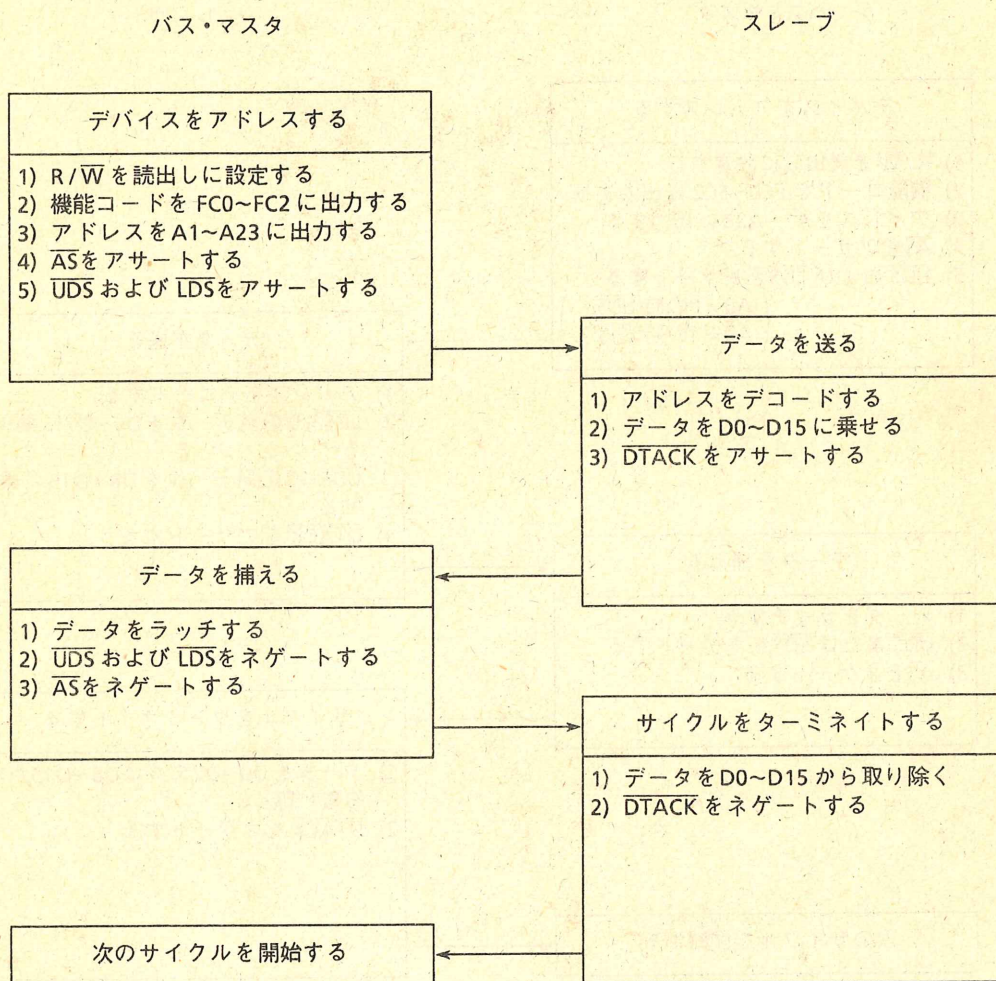



図4.5 ワード読出しサイクルのフローチャート

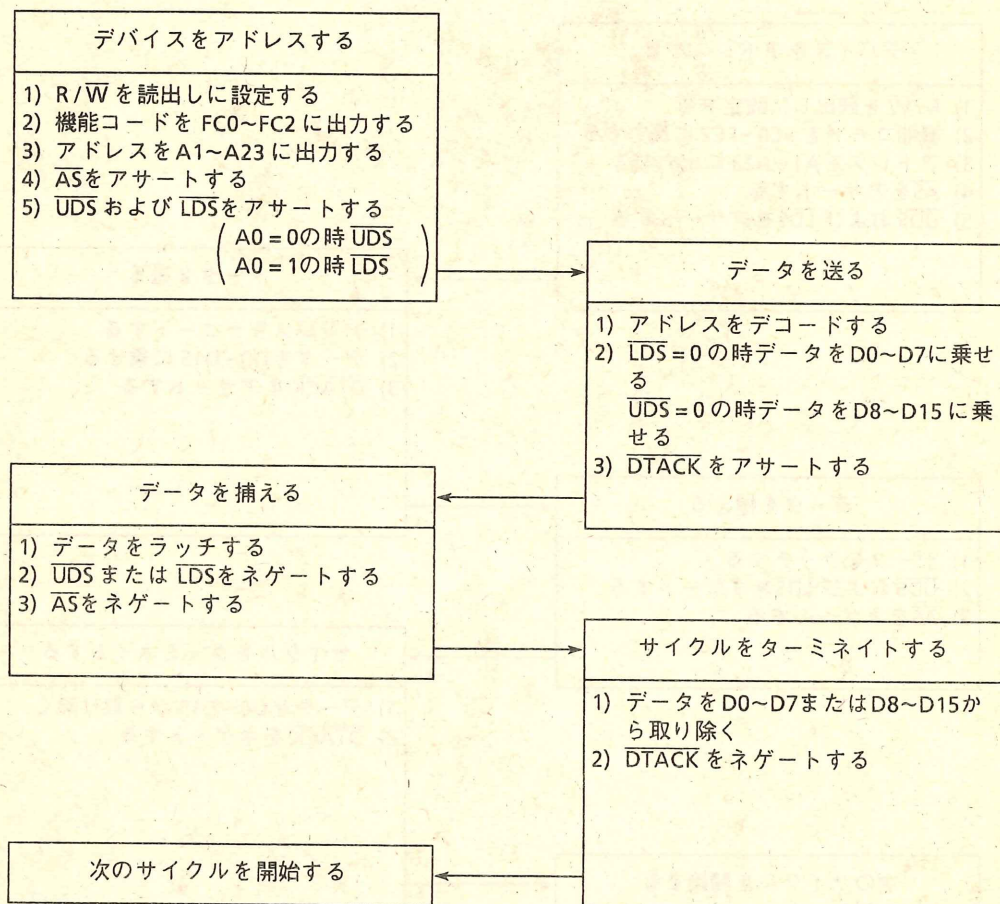



図4.6 バイト読出しサイクルのフローチャート

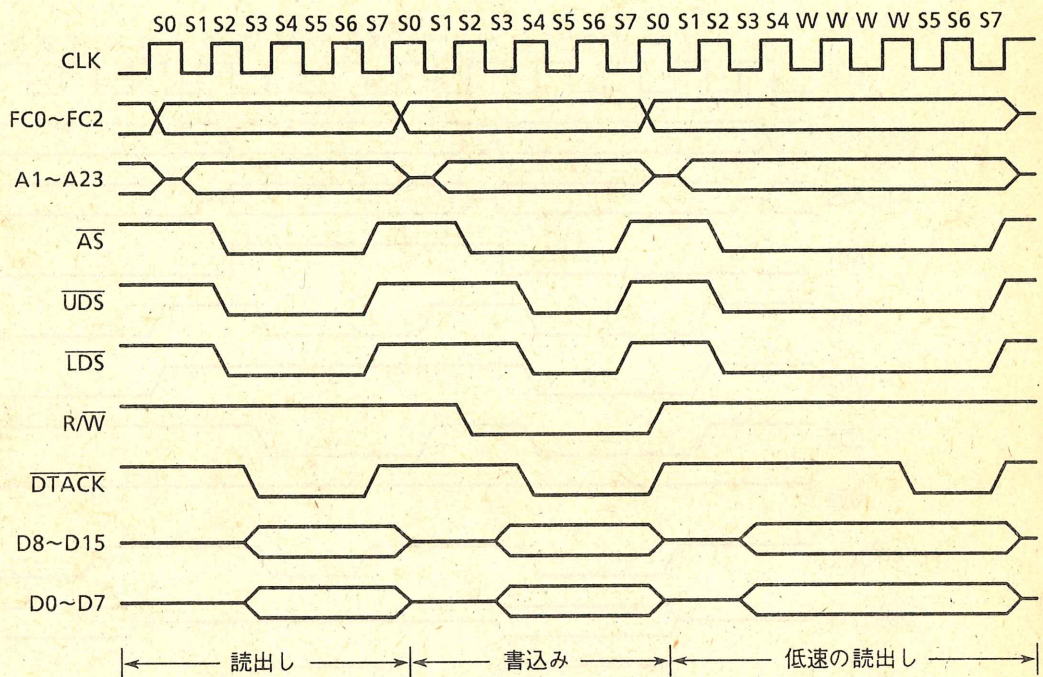



図4.7 読出しおよび書込みサイクルのタイミング

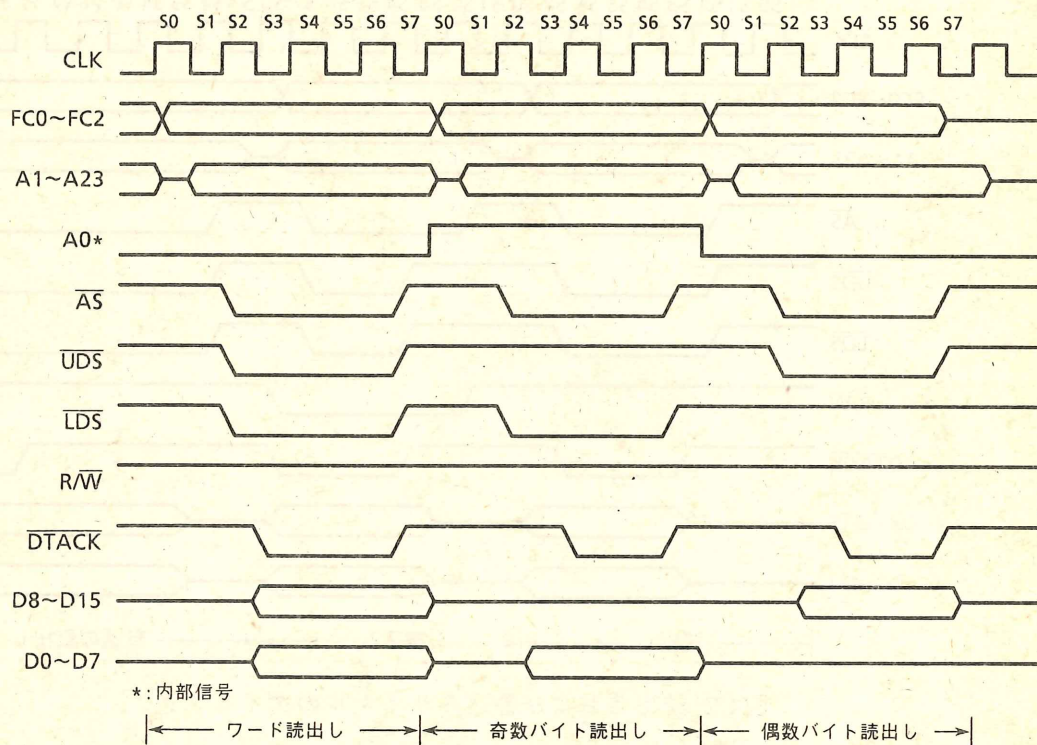



図4.8 ワードおよびバイトの読出しサイクルのタイミング

4.2.1.2 書込みサイクル

このサイクルでは、コア・プロセッサはデータをメモリまたは周辺デバイスへ送ります。コア・プロセッサはすべての場合においてバイト単位でデータを書込みます。命令がワードの動作を指定している場合は、コア・プロセッサは両方(上位および下位)のバイトを書込みます。バイト動作を指定している命令の場合は、コア・プロセッサは内部のA0ビットを使ってどちらのバイトを書込むかを決定し、そのバイトに必要なデータ・ストローブを出力します。すなわち、A0＝0の時は$\overline{\text{UDS}}$が、A0＝1の時は$\overline{\text{LDS}}$が出力されます。

ワードの書込みサイクルのフローチャートを図4.9に、バイトの場合を図4.10に示します。書込みサイクルのタイミングは図4.7に示されています。ワードおよびバイトの書込みサイクルのタイミングを図4.11に示します。

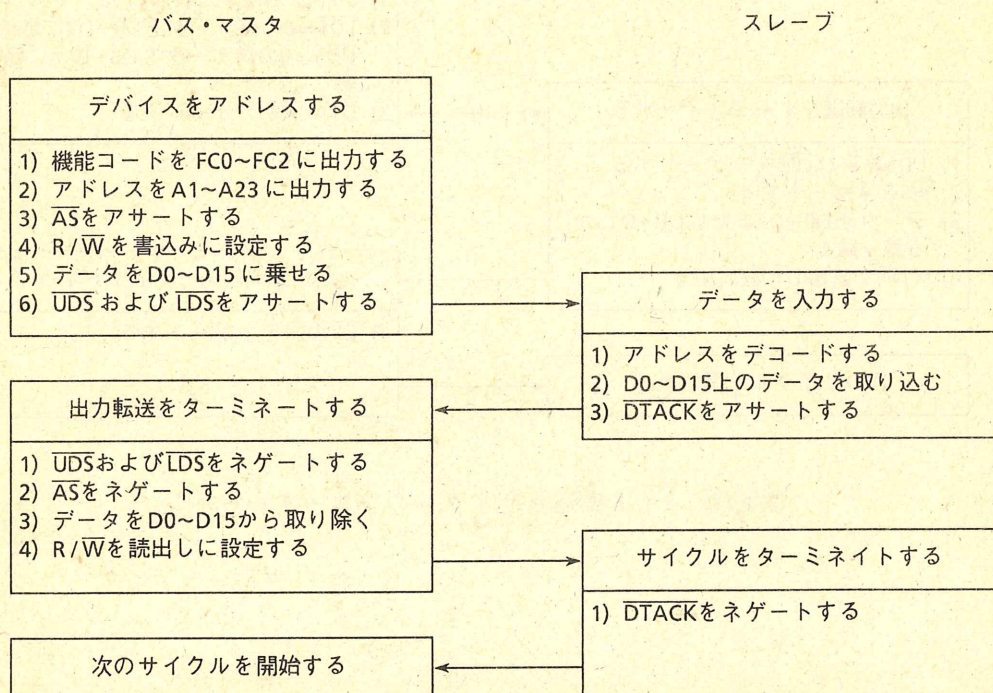



図4.9 ワード書込みサイクルのフローチャート

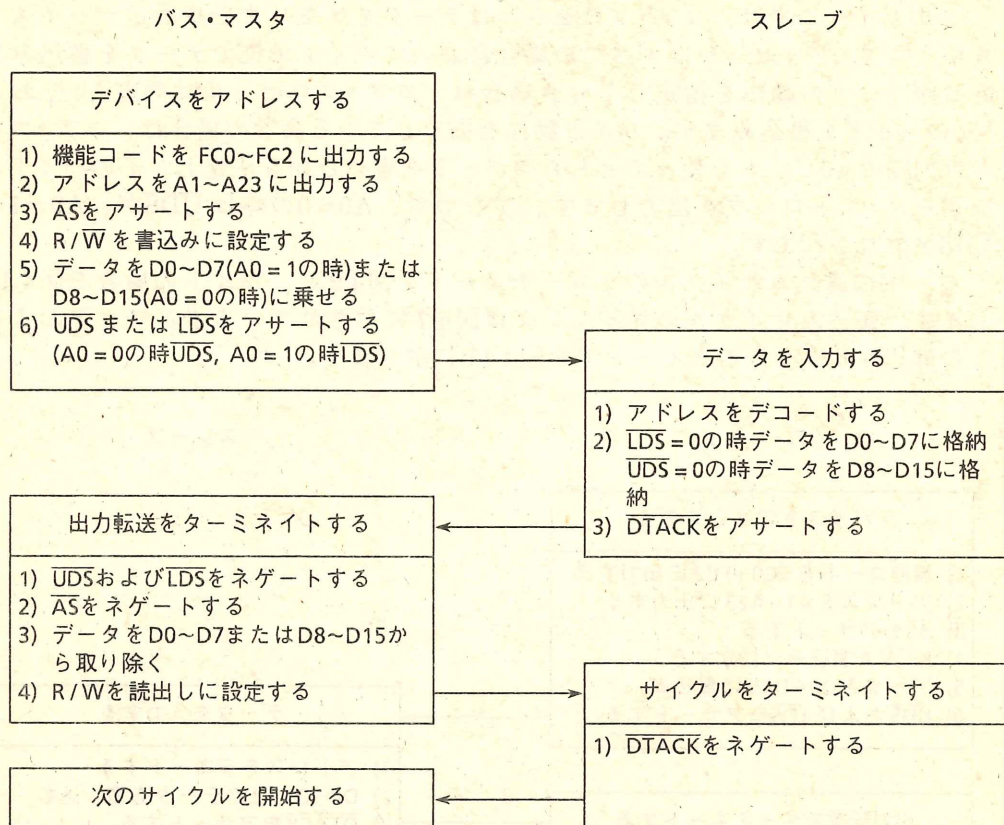



図4.10 バイト書込みサイクルのフローチャート

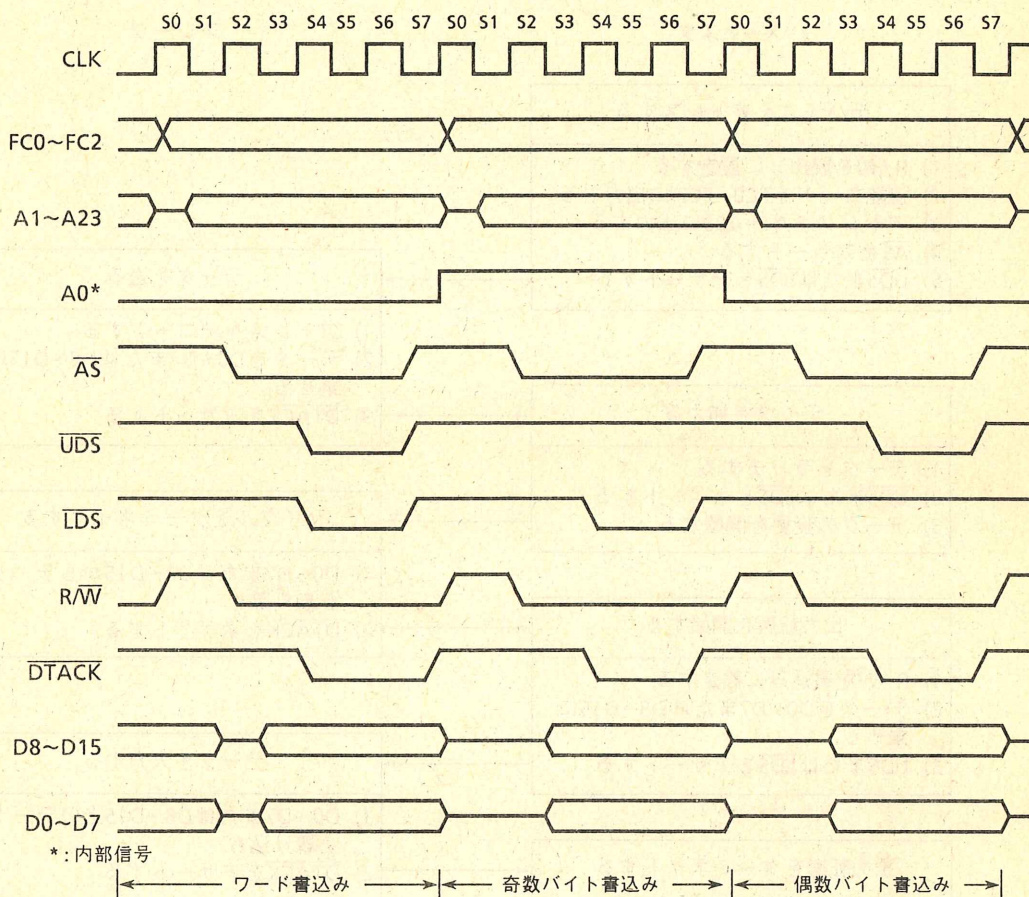



図4.11 ワードおよびバイトの書込みサイクルのタイミング

4.2.1.3 RMW(リード・モディファイ・ライト)サイクル

このサイクルではデータを読み、そのデータを演算論理ユニットの中で変更し、その結果を同じアドレスに書き込みます。TMP68301では、このサイクルは$\overline{AS}$がサイクル全体にわたってアサートされており、分離されないようになっています。テスト・アンド・セット(TAS)命令はこのサイクルを使って、マルチプロセッサ環境における通信が間違いなく行われるようにしています。RMWサイクルを使うのはTAS命令だけであり、TAS命令はバイト動作だけなのでRMWサイクルはすべてバイト動作です。RMWサイクルのフローチャートを図4.12に、タイミングを図4.13に示します。

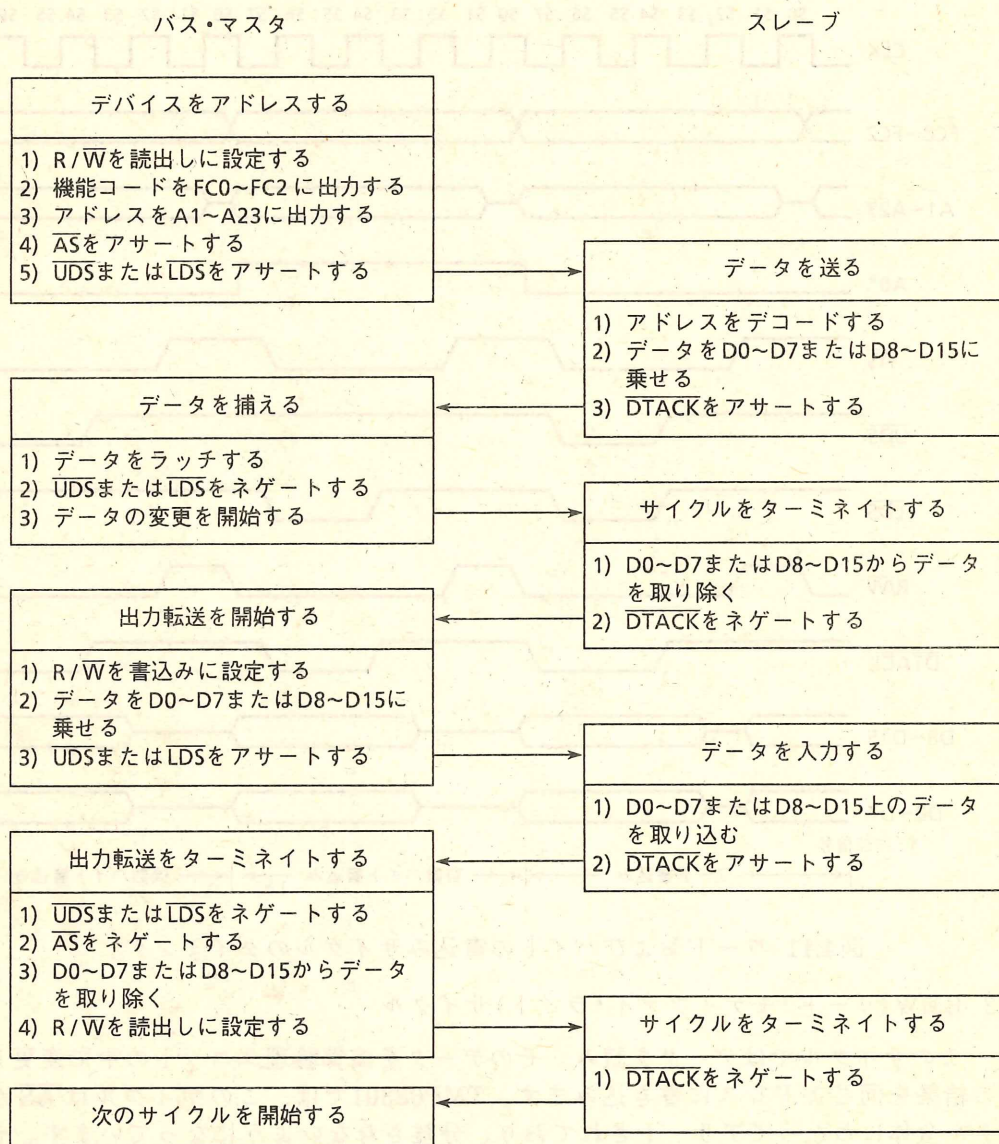



図4.12 RMWサイクルのフローチャート

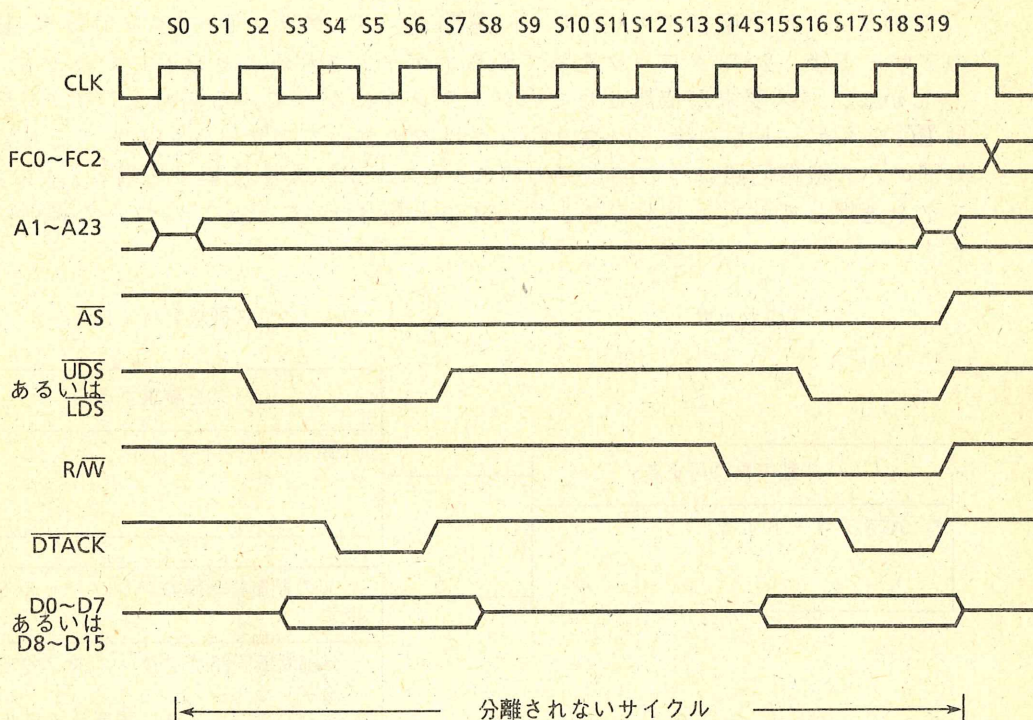



図4.13 RMWサイクルのタイミング

4.2.2 バス裁停

バス裁停というのはマスタ・タイプのデバイスによって使われ、バスの制御権の要求、許可、およびアクノリッジを行うための方式です。最も簡単な型では次の要素から構成されます。

1. バス制御権要求のアサート
2. バス使用許可(現在実行中のサイクルの終りで)の受信
3. バス制御権の獲得のアクノリッジ

単独デバイスからのバス要求動作のフローチャートを図4.14に、タイミングを図4.15に示します。この方式により、データ転送サイクル中のバス要求を処理することができます。

タイミング図では、バス・リクエスト(BR)はバス・グラント・アクノリッジ(BGACK)がアサートされた時点でネゲートされています。このタイプの動作はコア・プロセッサの他にバス・マスタになり得るデバイスが1個だけ存在するシステムの場合の動作です。バス・マスタになり得るデバイスが複数個存在するシステムでは、各デバイスからのバス要求信号はワイヤードORされてTMP68301に入ります。

そのようなシステムの場合、2個以上のデバイスが同時にバス要求を出す可能性があります。

このタイミング図はバス・グラント($\overline{BG}$)はバス・グラント・アクノリッジ($\overline{BGACK}$)のアサート後、2～3クロックたってからネゲートされることを示しています。

しかし、バス要求が依然としてペンディングになっている場合、コア・プロセッサは$\overline{BG}$をネゲートした後、2～3クロック以内に改めてアサートします。これによって外部のバス裁停回路はその時点でのバス・マスタがバスを放棄する前に，次のバス・マスタを選択しておくことができます。次に上記3つのステップについて詳細に説明します。

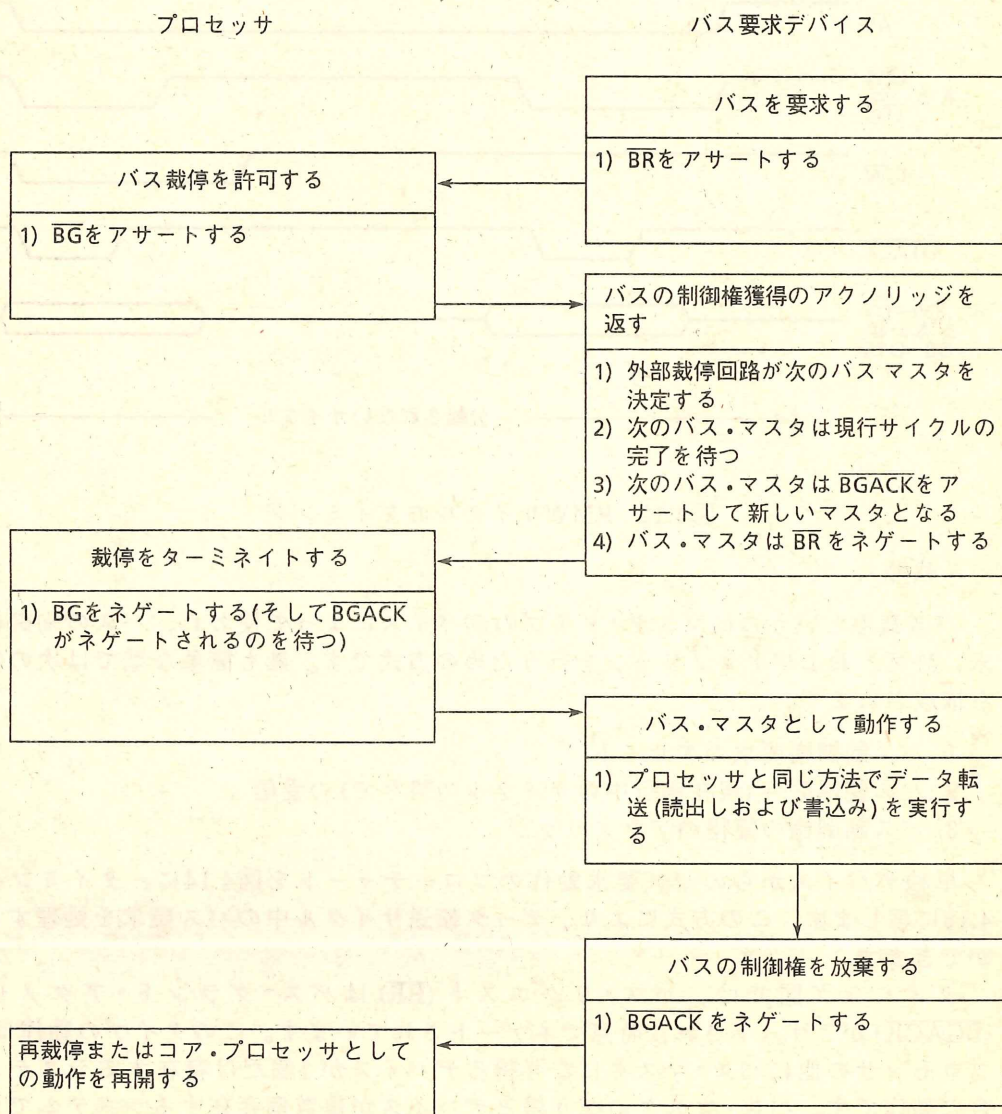



図4.14 バス裁停サイクルのフローチャート

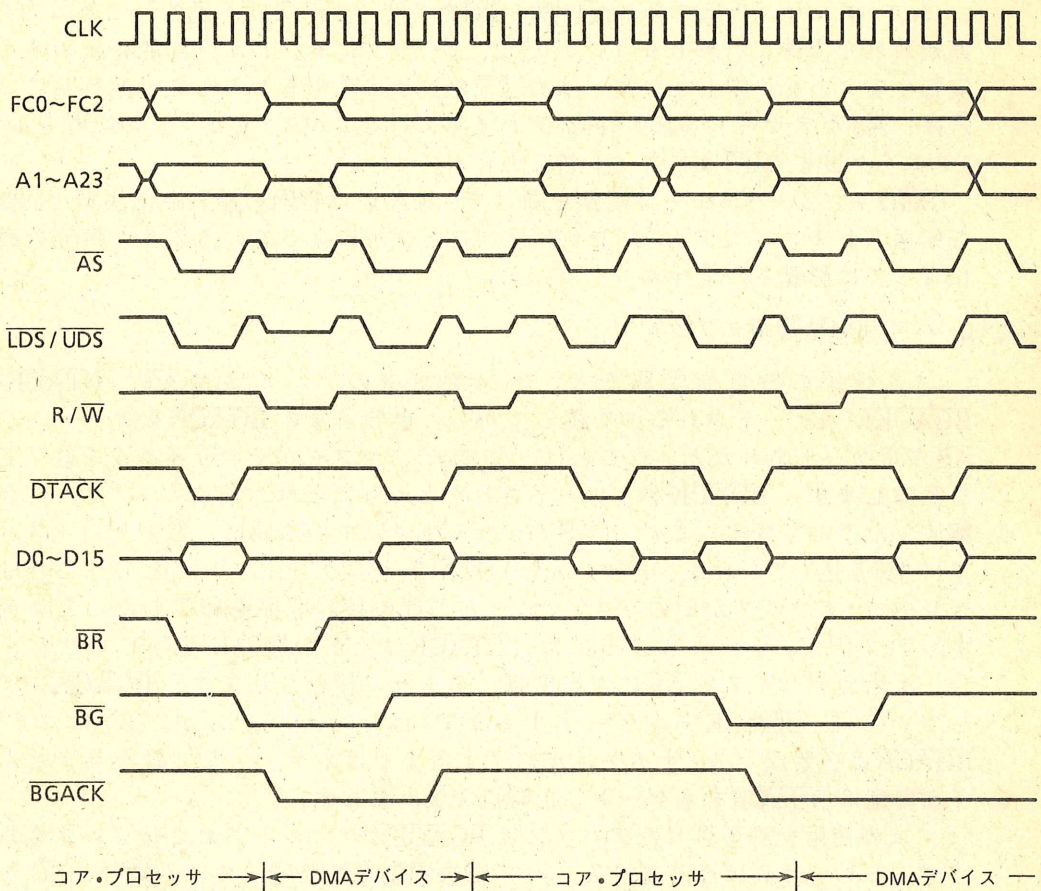



図4.15 バス裁停サイクル・タイミング

4.2.2.1 バス・リクエスト

バス・マスタになり得る外部デバイスは$\overline{BR}$信号をアサートすることによってバスを要求します。これはワイヤードOR(必ずしもオープン・コレクタ・デバイスから作られる必要はない)で、いくつかのデバイスが外部バスの制御権を要求していることを示します。コア・プロセッサは外部デバイスより優先順位が低く、その時点で開始されていたバス・サイクル終了後、バスを解放します。

バス・グラント・アクノリッジ($\overline{BGACK}$)が受信されないまま$\overline{BR}$がネゲートされた場合、コア・プロセッサは$\overline{BR}$のネゲーションを検出すると、処理を継続します。これによって、裁停回路がノイズに応答した場合でもプロセッサは正常に動作し続けます。

4.2.2.2 バス・グラントの受信

コア・プロセッサはできるだけ早く$\overline{\mathrm{BG}}$をアサートします。通常、これは内部での同期が取れた直後に行われます。ただし、コア・プロセッサが内部的に次のサイクルを実行することを決定していて、まだ$\overline{\mathrm{AS}}$をアサートするところまで進行していない場合は、$\overline{\mathrm{AS}}$がアサートされて(外部デバイスに対してバス・サイクルの実行を示す)から1クロック後まで$\overline{\mathrm{BG}}$はアサートされません。

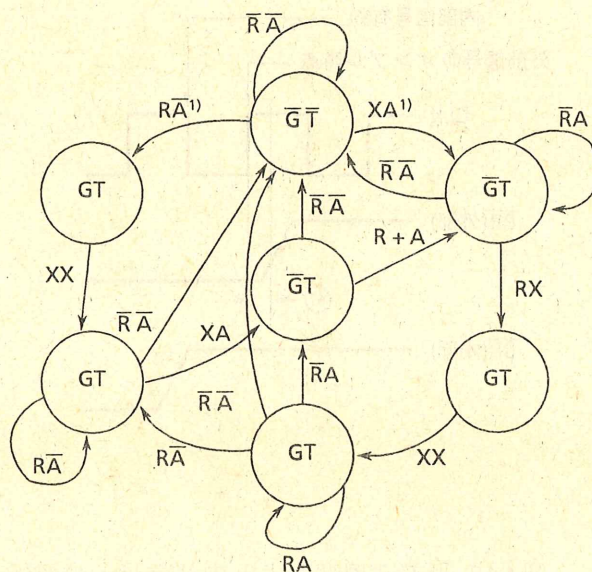
$\overline{\mathrm{BG}}$はデイジー・チェーン回路を通じて、あるいは特別の優先順位決定用回路を通して伝送されます。コア・プロセッサはプロトコルに合っている限り、外部の裁停方法によっては影響されません

4.2.2.3 バス制御権獲得のアクノリッジ

バス使用の許可を受取ると、バス要求中のデバイスは$\overline{\mathrm{AS}}$, $\overline{\mathrm{DTACK}}$および$\overline{\mathrm{BGACK}}$がネゲートされるのを待ってから、自分自身の$\overline{\mathrm{BGACK}}$をアサートします。$\overline{\mathrm{AS}}$がネゲートされたということは、前のバス・マスタがそのサイクルを終了したことを意味します。$\overline{\mathrm{BGACK}}$がネゲートされたということは、前のバス・マスタがバスを解放したことを意味します。($\overline{\mathrm{AS}}$がアサートされている間は、どのデバイスもサイクルに"割り込む"ことはできません。)$\overline{\mathrm{DTACK}}$がネゲートされたということは、前のスレーブ・デバイスが前のバス・マスタとの間の接続関係を終了したことを意味します。アプリケーションによっては、$\overline{\mathrm{DTACK}}$は上記の機能に関与しません。その場合、汎用のデバイスは$\overline{\mathrm{AS}}$にだけ関係するように接続されます。$\overline{\mathrm{BGACK}}$をアサートしている間($\overline{\mathrm{BGACK}}$をネゲートするまで)はそのデバイスがバス・マスタです。$\overline{\mathrm{BGACK}}$は必要なバス・サイクルが終了するまではネゲートしてはなりません。バスの制御権は$\overline{\mathrm{BGACK}}$をネゲートした時点で失われます。

バスの使用を許可されたデバイスは$\overline{\mathrm{BGACK}}$がアサートされた後にバス要求($\overline{\mathrm{BR}}$)はネゲートしなければなりません。バス要求($\overline{\mathrm{BR}}$)がまだ残っている場合、$\overline{\mathrm{BG}}$がネゲートされてから2～3クロック以内に改めて$\overline{\mathrm{BG}}$がアサートされます。4.2.3「バス裁停制御」の項を参照してください。コア・プロセッサは$\overline{\mathrm{BG}}$を再アサートする前に外部バス・サイクルを何も実行しません。

4.2.3 バス裁停制御

TMP68301の中のバス裁停制御ユニットは有限のステート機構によって実現されています。この機構のステート・ダイヤグラムを図4.16に示します。TMP68301への非同期入力信号はすべて同期されてから内部で使われます。この同期化は最大1システム・クロックの間に行われます。(非同期入力のセットアップ・タイム #47を満足していると仮定して)(図4.17参照)

R = 内部でのバス要求
A = 内部でのバス許可アクノリッジ
G = バス許可
T = バス制御信号に対する3ステート化
X = 任意

(注)

1) ステート機構はバスがS0またはS1ステートにある時は変化しない。
2) Tがアサートされていて$\overline{AS}$がネゲートされるとアドレス•バスはハイ•インピーダンス状態となる。

図4.16 TMP68301のバス裁停ユニットのステート•ダイヤグラム

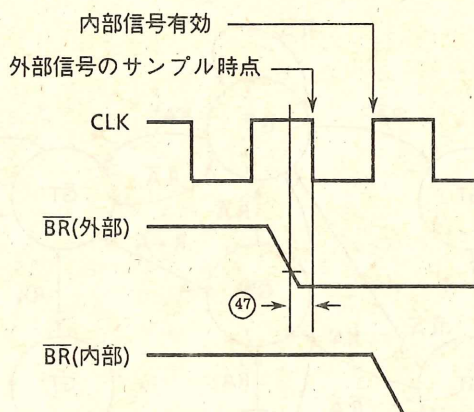



図4.17 外部非同期入力と内部信号との関係

入力信号はクロックの立下りでサンプルされ、次のクロックの立上りの後から内部的に有効となります。

図4.16に示すように、入力信号RおよびAは内部で入力信号の$\overline{BR}$および$\overline{BGACK}$がそれぞれ同期化された信号です。$\overline{BG}$出力はGと名付けられ、内部の3ステート制御信号はTと名付けられています。Tが真であれば、アドレス、データ、および制御の各バスは$\overline{AS}$がネゲートされた時点でハイ・インピーダンス状態になります。信号はすべて正論理(HIGHアクティブ)で示されています。(実際の信号のアクティブ・レベルの電圧の高低ではありません)。

ステートの変化(有効出力)は内部信号が有効になった後の次のクロックの立上りで発生します。

プロセッサ・バス・サイクル中のバス裁停シーケンスのタイミングを図4.18に示します。

バスが非アクティブ(すなわち、乗算命令などの内部動作を実行している時など)の状態でのバス裁停シーケンスを図4.19に示します。

TMP68301が既にバスサイクルを開始していて$\overline{AS}$がアサートされていない時点(S0)でバス要求があった場合、$\overline{BG}$は次のクロックの立上りではアサートされず、内部でのアサートに続く2番目のクロックの立上りまで遅らされます。このシーケンスを図4.20に示します。

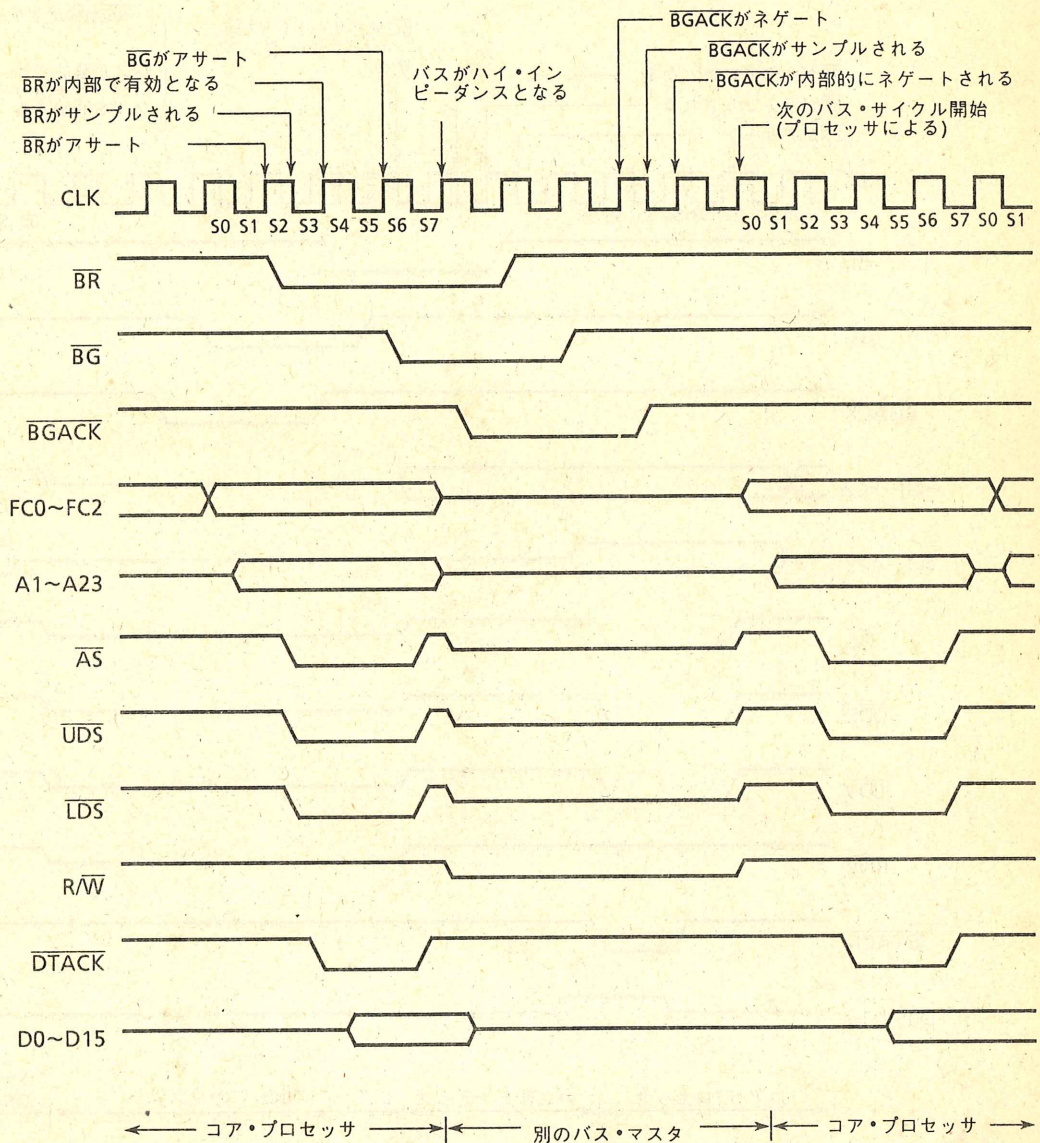



図4.18 プロセッサ・バス・サイクル中のバス裁停

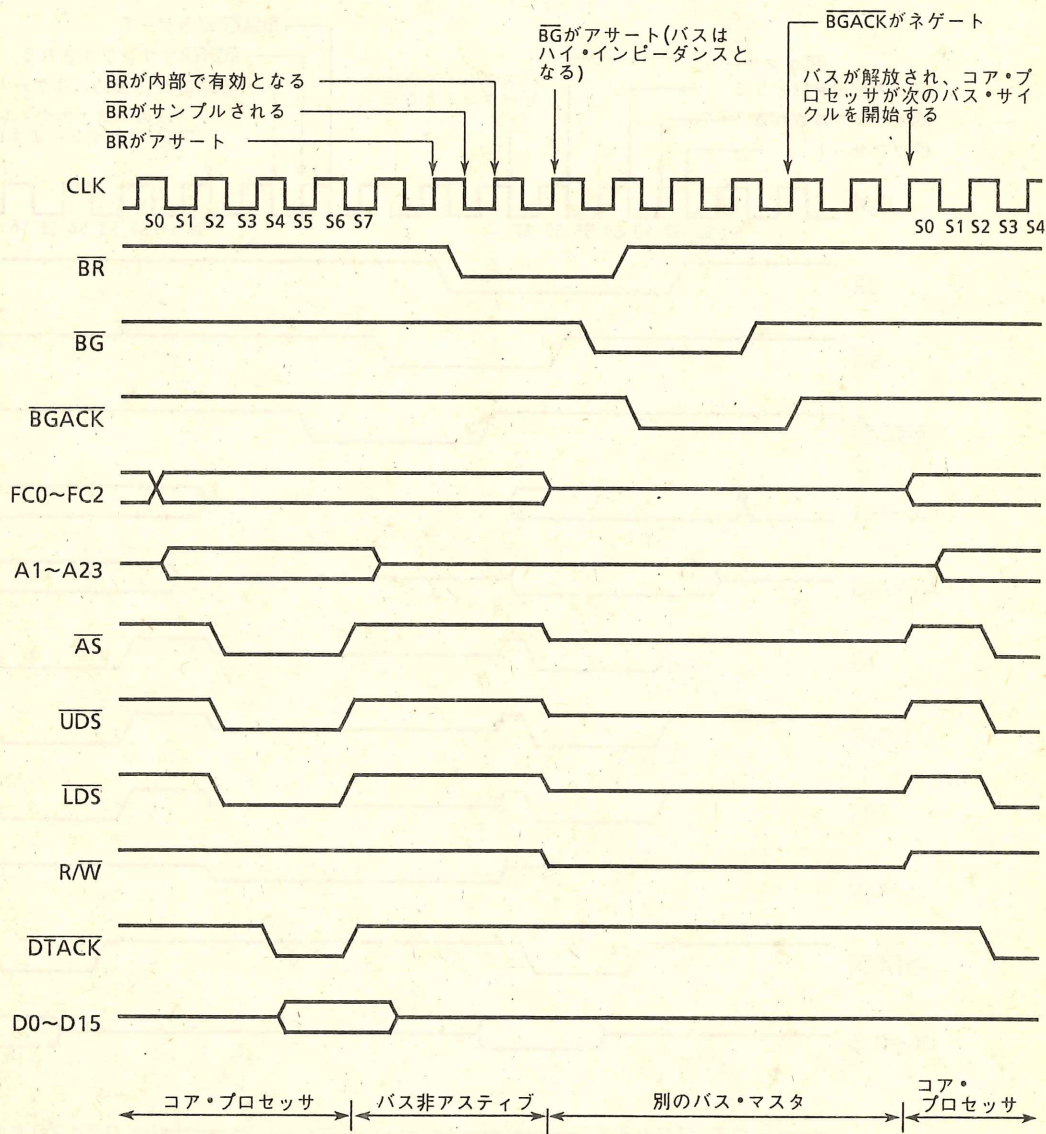



図4.19 バスが非アクティブ時のバス裁停

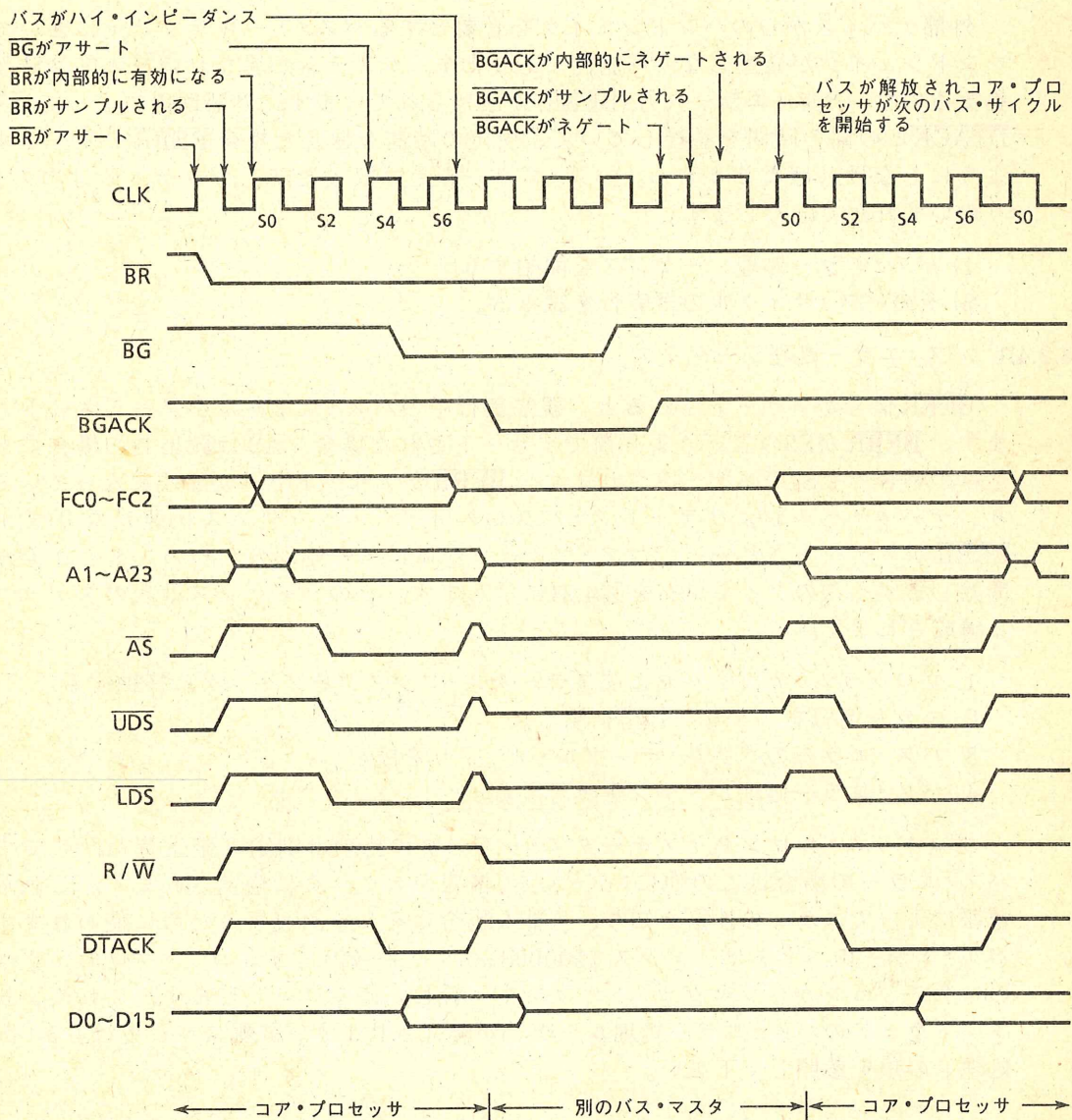



図4.20 コア・プロセッサ・バスサイクルの特殊なタイミングでバス要求が発生した場合

4.2.4 バス・エラーおよびホルト動作

外部デバイスからのハンドシェイクを必要とするバス・アーキテクチャにおいて、ハンドシェイクが発生しない可能性があります。システムが異なれば最大応答時間も異なるので、バス・エラー入力($\overline{\text{BERR}}$)が設けられています。外部回路によって$\overline{\text{AS}}$と$\overline{\text{DTACK}}$との間の時間を監視していて、所定の時間を越えた場合$\overline{\text{BERR}}$アサートするようにしなければなりません。バス・エラーを受け取った時、プロセッサは次の2通りのいずれかで応答します。

1) バス・エラー処理シーケンスを開始する。
2) そのバス・サイクルの再実行を試みる。

4.2.4.1 バス・エラー処理シーケンス

$\overline{\text{BERR}}$信号がアサートされると、現在実行中のバス・サイクルがターミネートされます。$\overline{\text{BERR}}$がS2の立下りより前でアサートされた場合、$\overline{\text{AS}}$は読出しの場合でも書込みの場合でもS7でネゲートされます。$\overline{\text{BERR}}$がアサートされたままになっている限り、データ・バスおよびアドレス・バスはハイ・インピーダンス状態になります。$\overline{\text{BERR}}$がネゲートされると、コア・プロセッサはエラー処理のためのスタック操作を開始します。このタイミングを図4.21に示します。このシーケンスは次のステップから構成されます。

1. プログラム・カウンタおよびステータス・レジスタをスタックに格納する。
2. エラー情報をスタックに格納する。
3. バス・エラーのベクタ・テーブル・エントリを読む。
4. バス・エラー処理ルーチンを実行する。

プログラム・カウンタとステータス・レジスタの格納は割込み発生時と同じです。バス・エラーの場合はこの他にいくつかの情報がスタックに格納されます。これらの情報によってエラーの性格を知り、可能な場合はそれを回復するために使われます。バス・エラーのベクタはアドレス$000008(ベクタ番号2)です。コア・プロセッサはこのロケーションからプログラム・カウンタへ新しい値をロードします。これにより、ソフトウェアのバス・エラー処理ルーチンが実行されます。詳細については5.2「例外処理」の項を参照して下さい。

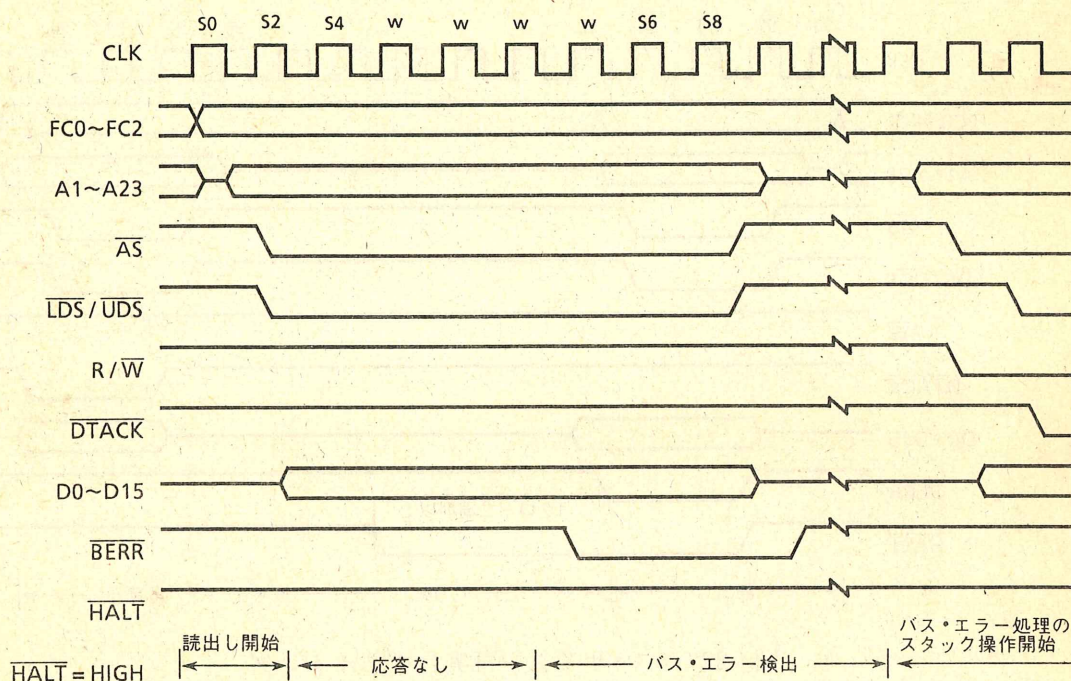



図4.21 バス・エラーのタイミング

4.2.4.2 バス・サイクルの再実行

バス・サイクル実行中にBERRがアサートされ、同時にHALTもアサートされている時、コア・プロセッサは再実行シーケンスに入ります。図4.22にこのタイミングを示します。

コア・プロセッサはそのバス・サイクルをターミネートしてから、アドレス・バスおよびデータ・バスをハイ・インピーダンス状態にします。HALTが外部デバイスによってネゲートされるまでは、コア・プロセッサは停止したままであり、次のバス・サイクルの実行に移りません。HALTがネゲートされると、コア・プロセッサは同じアドレス、同じデータ（書込み動作の場合）同じ制御信号を使って、前のバス・サイクルを再実行します。BERRはHALTがネゲートされるより少くとも、1クロック前にネゲートされなければなりません。

(注)

コア・プロセッサはRMWサイクルの再実行は行いません。この制限はRMWサイクル全体が正しく実行され、テスト・アンド・セット命令の書込み動作がASを解放せずに実行されることを保証するために設けられています。BERRとHALTがRMWサイクル中にアサートされた場合、バス・エラー処理動作が実行されます。

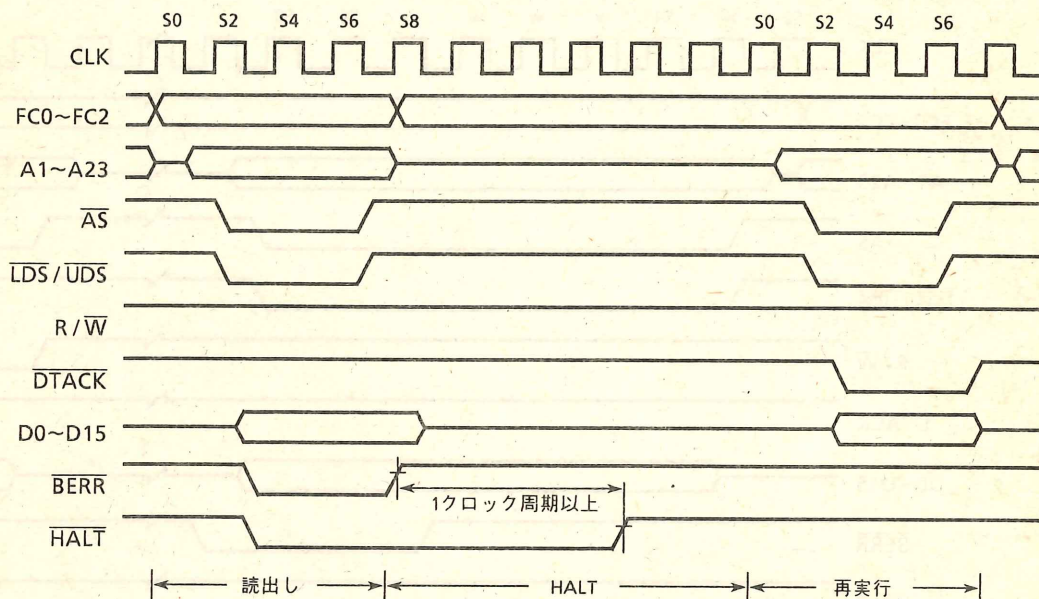



図4.22 バス・サイクル再実行のタイミング

4.2.4.3 バス・エラーを伴わないホルト動作

TMP68301への$\overline{\text{HALT}}$を使って6800のホルト機能と同様に停止/実行/シングル・ステップを実行することができます。停止および実行のモードはその名の通りで$\overline{\text{HALT}}$がアサートされている時、コア・プロセッサは“停止”(何もしない)しており、$\overline{\text{HALT}}$がアサートされていない時はコア・プロセッサは“実行”(何かを行っている)状態にあります。

シングル・ステップ・モードは$\overline{\text{HALT}}$を適切なタイミングで操作することによって得られます。これはコア・プロセッサが1つのバス・サイクルを開始するまで"実行"モードにしておき、その後"停止"モードに切り変えることによって行われます。このシングル・ステップ・モードによって、コア・プロセッサの動作をバス・サイクルの単位で追うことができます。

シングル・ステップ動作を正しく行わせるのに必要なタイミングの詳細を図4.23に示します。シングル・ステップ・モードを使ってデバッグする場合、$\overline{\text{BERR}}$と$\overline{\text{HALT}}$の干渉にも注意する必要があります。

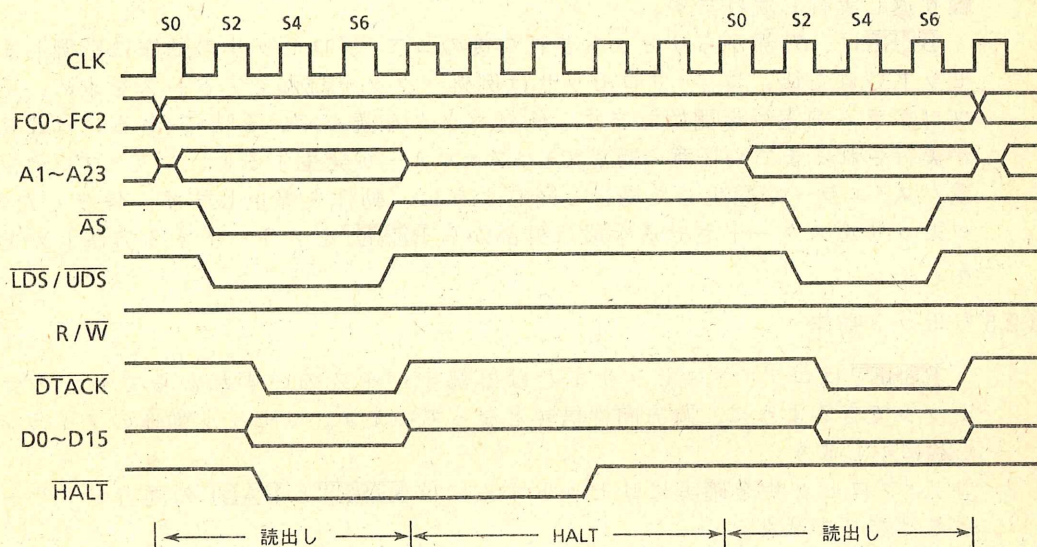



図4.23 HALTのタイミング

HALTがアサートされたことを、コア・プロセッサが認識した後、バス・サイクルを完了すると、ほとんどの3ステート・ピンがハイ・インピーダンス状態になります。この中には次のピンが含まれます。

1. アドレス・バス(A1~A23)
2. データ・バス(D0~D15)

これはバス・サイクル動作を正しく再実行させるために必要です。

コア・プロセッサがホルト要求を認めている間においても、バス裁停は普通に行われます。すなわち、ホルト動作はバス裁停に影響しません。バスから制御信号を取り除くのはバス裁停の機能です。

ホルト機能とハードウェア・トレース機能により、ハードウェア・デバッガがバス・サイクルを1つずつ、あるいは命令を1つずつトレースすることができます。

これらの機能をソフトウェア・デバッグ・パッケージと併用することによって柔軟性のあるデバッグが可能となります。

4.2.4.4 二重バス・エラー

バス・エラーが発生して、コア・プロセッサがデバイスの状態を含むいくつかの情報をスタックに格納しようとしている時にふたたびバス・エラーが発生した場合、二重バス・エラーとなります。この場合、コア・プロセッサは停止します。一度バス・エラーが発した後、次の命令が実行される前にふたたびバス・エラーが発生するとすべての二重バス・エラーとなります。

再実行のバス・サイクルが依然としてエラー状態であっても、それはバス・エラーとは見なされず、したがって二重バス・エラーにはなりません。すなわち、外部ハー

ドウェアがエラー状態を解除しない限り、コア・プロセッサは同じバス・サイクルを繰り返し実行し続けます。

$\overline{BERR}$は、外部からリセットされた後のコア・プロセッサの動作に影響します。リセットされた後、コア・プロセッサは例外ベクタを読んでアドレスを求め、そこからプログラムの実行を開始します。例外ベクタを読んでいる時に(あるいは最初の命令が実行されるまでの任意の時点で)バス・エラーが発生すると、コア・プロセッサは二重バス・エラーが発生した場合と同じように、動作を停止します。停止したコア・プロセッサをスタートさせる手段は外部から$\overline{RESET}$をアサートする方法しかありません。

4.2.5 リセット動作

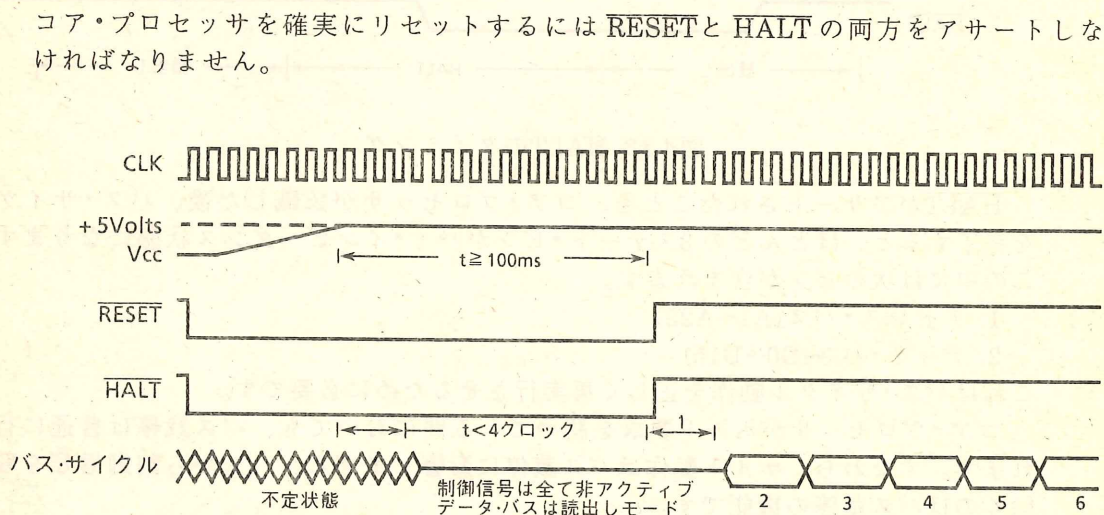
$\overline{RESET}$はコア・プロセッサまたは外部デバイスのいずれからでもシステムをリセットできるように、双方向性信号となっています。リセット動作のタイミングを図4.24に示します。
コア・プロセッサを確実にリセットするには$\overline{RESET}$と$\overline{HALT}$の両方をアサートしなければなりません。

(注)
(1) 内部スタート・アップ・タイム
(2) SSPの上位を読込む
(3) SSPの下位を読込む
(4) PCの上位を読込む
(5) PCの下位を読込む
(6) 最初の命令フェッチ

図4.24 リセット動作のタイミング

$\overline{\mathrm{RESET}}$と$\overline{\mathrm{HALT}}$が外部デバイスによってアサートされると、システム全体(プロセッサを含む)のリセットとして確認されます。コア・プロセッサはこれに応答してリセットのベクタ・テーブル・エントリ(アドレス$000000にあるベクタ番号0)を読み、それをスーパバイザ・スタック・ポインタ(SSP)にロードします。次にベクタ・テーブル・エントリ番号1(アドレス$000004)を読んでコア・プログラム・カウンタ(PC)にロードします。コア・プロセッサはステータス・レジスタの割込みレベル・フィールドを7にイニシャライズします。上記以外のレジスタはリセット・シーケンスによっては影響されません。リセット命令が実行されると、コア・プロセッサは$\overline{\mathrm{RESET}}$を124クロックの間アサートします。この場合、コア・プロセッサはシステムの自分以外のデバイスをリセットしようとしています。したがって、コア・プロセッサの内部状態は影響されません。コア・プロセッサの内部のレジスタおよびステータス・レジスタはいずれもリセット命令の実行によっては影響されません。$\overline{\mathrm{RESET}}$が接続されているデバイスはすべてリセット命令の終了時点でリセットされていなければなりません。

$\overline{\mathrm{RESET}}$と$\overline{\mathrm{HALT}}$を10クロック・サイクルの間アサートすると、TMP68301はリセットされます。ただし、Vccを最初にTMP68301に印加する時は、外部回路によって100ms以上アサートされなければなりません。

4.3 $\overline{\mathrm{DTACK}}$, $\overline{\mathrm{BERR}}$, および$\overline{\mathrm{HALT}}$の関係

再実行またはバス・エラー状態でバス・サイクルの終了を正しく制御するために、$\overline{\mathrm{DTACK}}$, $\overline{\mathrm{BERR}}$および$\overline{\mathrm{HALT}}$のアサートおよびネゲートはTMP68301のクロックの立上りで行う必要があります。これによって、2つの信号が同時にアサートされた場合、両方の信号に対するセットアップ・タイム(#47)は同じバス・ステートの間で満足されます。

上記、または上記と等価な手段を外部回路で講じておく必要があります。パラメータ#48($\overline{\mathrm{BERR}}$の立下りから$\overline{\mathrm{DTACK}}$の立下りまで)はこの動作を非同期システムで保証するために設けられており、上記の条件が満足されている場合は無視して構いません。

望ましいバス・サイクル・ターミネーションの条件を次に要約します。(ケース番号は表4.4を参照してください)。

ノーマル・ターミネーション: $\overline{\mathrm{DTACK}}$が最初に発生する(ケース1)。

ホルト・ターミネーション: $\overline{\mathrm{BERR}}$がネゲートされたままで、$\overline{\mathrm{DTACK}}$と同時またはそれ以前に$\overline{\mathrm{HALT}}$がアサートされる。(ケース2, 3)

バス・エラー・ターミネーション: $\overline{\mathrm{BERR}}$が$\overline{\mathrm{DTACK}}$の代りに、$\overline{\mathrm{DTACK}}$と同時、あるいは$\overline{\mathrm{DTACK}}$に先立ってアサートされる(ケース4)。$\overline{\mathrm{BERR}}$のネゲートは、$\overline{\mathrm{DTACK}}$のネゲートと同時またはそれより後に行われる。

再実行ターミネーション: $\overline{\text{HALT}}$および$\overline{\text{BERR}}$が$\overline{\text{DTACK}}$の代わりに、$\overline{\text{DTACK}}$と同時またはその前にアサートされる(ケース6, 7)。$\overline{\text{BERR}}$は$\overline{\text{HALT}}$より少なくとも1クロック前にネゲートされなければならない。ケース5は$\overline{\text{BERR}}$が$\overline{\text{HALT}}$に先行してもよいことを示しており、それによって完全に非同期のアサートが許されます。

表4.4は制御信号の各種の組合せによるバス•サイクルのターミネーションの詳細を示しています。いくつかの状態での制御信号のネゲート条件を表4.5に示します($\overline{\text{DTACK}}$はすべての場合において普通にネゲートされると仮定しています。$\overline{\text{DTACK}}$と$\overline{\text{BERR}}$を両方共$\overline{\text{AS}}$のネゲートと同時にネゲートするのが最良の方法です)。

表4.4 $\overline{\mathrm{DTACK}}$, $\overline{\mathrm{BERR}}$および$\overline{\mathrm{HALT}}$のアサーションの結果

ケース番号	制御信号	立上りでアサートされるステート		結果
		N	N+2	
1	$\overline{\mathrm{DTACK}}$ $\overline{\mathrm{BERR}}$ $\overline{\mathrm{HALT}}$	A NA NA	S X X	ノーマル・サイクル・ターミネートおよび継続。
2	$\overline{\mathrm{DTACK}}$ $\overline{\mathrm{BERR}}$ $\overline{\mathrm{HALT}}$	A NA A	S X S	ノーマル・サイクル・ターミネートおよび停止。$\overline{\mathrm{HALT}}$が取り除かれると継続。
3	$\overline{\mathrm{DTACK}}$ $\overline{\mathrm{BERR}}$ $\overline{\mathrm{HALT}}$	NA NA A	A NA S	ノーマル・サイクル・ターミネートおよび停止。$\overline{\mathrm{HALT}}$が取り除かれると継続。
4	$\overline{\mathrm{DTACK}}$ $\overline{\mathrm{BERR}}$ $\overline{\mathrm{HALT}}$	X A NA	X S NA	ターミネートした後バス・エラー・トラップ。
5	$\overline{\mathrm{DTACK}}$ $\overline{\mathrm{BERR}}$ $\overline{\mathrm{HALT}}$	NA A NA	X S A	ターミネートした後再実行。
6	$\overline{\mathrm{DTACK}}$ $\overline{\mathrm{BERR}}$ $\overline{\mathrm{HALT}}$	X A A	X S S	ターミネートした後, $\overline{\mathrm{HALT}}$が取り除かれると再実行。
7	$\overline{\mathrm{DTACK}}$ $\overline{\mathrm{BERR}}$ $\overline{\mathrm{HALT}}$	NA NA A	X A S	ターミネートした後, $\overline{\mathrm{HALT}}$が取り除かれると再実行。

N : 偶数のバス・ステート番号(たとえばS4, S6など)

A : このバス・ステートでアサートされる。

NA : このバス・ステートでアサートされない。

X : 任意

S : 信号が前のステートでアサートされていて、このステートも継続してアサートされている。

表4.5 $\overline{\mathrm{BERR}}$ と $\overline{\mathrm{HALT}}$ のネゲーションの結果

表4.4での ターミネートの状態	制御信号	立上りでアサートされるステート		結果 — 次のサイクル
		N	N+2	
バスエラー	$\overline{\mathrm{BERR}}$	● または	●	バス・エラー・トラップ発生
	$\overline{\mathrm{HALT}}$	● または	●	
再実行	$\overline{\mathrm{BERR}}$	● または	●	イリーガル・シーケンス:通常ベクタ番号0へトラップする。
	$\overline{\mathrm{HALT}}$	●		
再実行	$\overline{\mathrm{BERR}}$	●		バス・サイクルの再実行
	$\overline{\mathrm{HALT}}$		●	
ノーマル	$\overline{\mathrm{BERR}}$	●		次のサイクルの開始をのばすことができる。
	$\overline{\mathrm{HALT}}$	● または	●	
ノーマル	$\overline{\mathrm{BERR}}$		●	次のサイクルが開始された場合バス・エラーとしてターミネートされる。
	$\overline{\mathrm{HALT}}$	● または無し		

● : このバス・サイクルでネゲートされる。

例A : 使われていないアドレス空間へのアクセスをターミネートするためにウォッチドッグ・タイマを設け、タイムアウト時に $\overline{\mathrm{DTACK}}$ と $\overline{\mathrm{BERR}}$ を同時にアサートします。(ケース4)

例B : RAMの内容のエラー検出を行っているシステム。
この場合、次の3通りの方法があります。

(a) データのチェックが済むまで $\overline{\mathrm{DTACK}}$ のアサートを遅らせ、エラー検出された場合は $\overline{\mathrm{BERR}}$ と $\overline{\mathrm{HALT}}$ を同時にアサートしてサイクルを再実行させる。(ケース6)
エラーが検出されなかった場合は $\overline{\mathrm{DTACK}}$ を返すようにする。(ケース1)

(b) データの検証が済むまで $\overline{\mathrm{DTACK}}$ のアサートを遅らせ、エラーが検出された場合 $\overline{\mathrm{BERR}}$ を $\overline{\mathrm{DTACK}}$ と同時に返すようにする。(ケース4)

4.4 非同期動作と同期動作

4.4.1 非同期動作

システム・レベルでのクロック周波数に対する独立性を実現するために、TMP68301は非同期方式で使うことができます。この場合、データ転送を制御するためにバス・ハンドシェーク信号($\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{HALT}$, および$\overline{VPA}$)を使うことが必要です。この方法を使う場合、$\overline{AS}$がバス・サイクルの開始を示し、$\overline{UDS}$/$\overline{LDS}$が書込みサイクル中でデータが有効であることを示すために使われます。スレーブ・デバイス(メモリまたは周辺デバイス)は読出しサイクルにおいては要求されたデータをデータ・バス上に乗せ、書込みサイクルにおいてはデータを取り込んだ後$\overline{DTACK}$をアサートすることによって応答し、バス・サイクルをターミネートします。スレーブ・デバイスが応答しないか、アクセスが無効なものであった場合、外部回路によって$\overline{BERR}$または、$\overline{BERR}$と$\overline{HALT}$をアサートして、そのバス・サイクルを打ち切るか再実行するかを指示します。

$\overline{DTACK}$は読出しサイクルにおいてはスレーブ・デバイスからのデータが有効になる前にアサートすることが許されます。その場合$\overline{DTACK}$が先行し得る時間の長さはパラメータ#31で規定されており、どんな非同期システムにおいてもコア・プロセッサの中に正しいデータが取り込まれるためには、この規定値が満足されなければなりません。ここでASがアサートされてから$\overline{DTACK}$がアサートされるまでの時間の許容最大値は規定されていないことに注意してください。これは$\overline{DTACK}$が認知されるまでプロセッサが1クロックのウェイト・ステートを挿入するからです。

4.4.2 同期動作

$\overline{DTACK}$および他の非同期入力信号を発生するのにシステム・クロックを使うシステムが構成できるようにするために、非同期入力用のセットアップ・タイムとしてパラメータ#47が規定されています。この値が$\overline{DTACK}$などの入力で満足されている場合プロセッサはシステム・クロックの次の立下りでその信号を認知することが保証されます。しかし、その逆は成立しません。すなわち、入力信号がセットアップ・タイムを満足していないからといって、必ず認知されないというわけはありません。この他に、$\overline{DTACK}$が立下りで認められた場合、パラメータ#27で規定されているセットアップ・タイムを満足していれば次の立下りで有効なデータがコア・プロセッサ内部に取り込まれます。(読出しサイクルの場合)。上記の条件が満足されていれば、パラメータ#31は無視して構いません。S4の立下りより前に$\overline{DTACK}$がアサートされたは場合(セットアップ時間を満足して)、ウェイト・ステートは挿入されず、バス・サイクルは最高速の4クロックで実行されます。

非同期システムで$\overline{BERR}$が$\overline{DTACK}$の後でアサートされる場合に正しく動作することを保証するために、$\overline{BERR}$は$\overline{DTACK}$が認知されたクロックの1つ後のクロックの立下り時点より前にパラメータ#27のセットアップ時間が満足してアサートされなければなりません。このセットアップ・タイムは正常な動作を保証するために重要なパラメータで、これが満足されていない場合、TMP68301が誤動作する可能性があります。

(注)

アクティブなバス・サイクルの間、$\overline{BERR}$はS2から始まるすべてのクロックの立下りでサンプリングされます。$\overline{DTACK}$はS4から始まるすべてのクロックの立下りでサンプルされ、読出しの場合はS6の立下りでデータが取り込まれます。バス・サイクルは$\overline{DTACK}$がアサートされないまま、$\overline{BERR}$がアサートされた時はS9でターミネートされます。それ以外はS7でターミネートされます。

第5章 処理ステート

この章では命令の実行に伴う通常の処理以外のTMP68301の動作について説明します。ステータス・レジスタのスーパバイザ部分のビット(スーパバイザ・ステート・ビット，トレース・モード・ビットおよび割込みマスク)の機能を説明します。最後に、例外状態発生時のコア・プロセッサのメモリ参照シーケンスおよび動作について説明します。

TMP68301は常に3つの処理ステート、すなわち、ノーマル，例外，またはホルトのいずれかのステートにあります。ノーマル処理ステートは命令実行時のステートで、メモリ参照は命令およびオペランドのフェッチおよび結果の格納のために行われます。ノーマルステートの特殊ケースがストップ・ステートで、STOP命令が実行されるとこのステートに入ります。このステートではメモリ参照は行なわれません。

例外処理ステートは、割込み，トラップ命令，トレース動作等の例外状態によって発生します。例外は命令によって内部的に発生させることができ、また命令の実行中に生じる異常状態によっても発生します。外部的には例外処理は割込み，バス・エラー，またはリセットによって要求されます。例外処理はコア・プロセッサが異常状態に対して効率よく処理の流れを切換えられるように設計されています。

ホルト処理ステートは重大なハードウェア故障を意味します。たとえば、バス・エラーの処理中にふたたびバス・エラーが発生した場合、コア・プロセッサはシステムが異常になったと判断して停止します。停止したコア・プロセッサを再スタートさせる手段は外部から$\overline{\text{RESET}}$をアサートする以外にありません。ホルト・ステートはストップ・ステートとは異なるものであることに注意してください。

5.1 特権ステート

コア・プロセッサはユーザ・ステートまたはスーパバイザ・ステートのいずれかの特権ステートで動作します。ステートによって、どの動作が正しいかが決定され、外部のメモリ管理デバイスはステートを使ってアクセスの制御およびアドレス変換を行います。また命令でスタック・ポインタとしてSSPを使うか、USPを使うかがこのステートで決まります。

特権ステートはコンピュータ・システムの機密保護のために設けられたメカニズムの1つです。プログラムは自分専用のコード領域とデータ領域だけにアクセスすべきであり、不要な情報および変更してはいけない情報へのアクセスは禁止されています。

特権ステートのメカニズムでは、大部分のプログラムがユーザ・ステートで実行されるようにすることによって、コンピュータ・システムを保護します。ユーザ・ステートでは、アクセスが管理され、システムの外の部分への影響が制限されます。オペレーティング・システムはスーパバイザ・ステートで稼働し、すべてのリソースにアクセスでき、ユーザ・ステートのプログラムを管理するタスクを実行します。

5.1.1 スーパバイザ・ステート

スーパバイザ・ステートはユーザ・ステートより優先順位が高いステートです。命令の実行に対して、スーパバイザ・ステートはステータス・レジスタのSビットによって決定されます。ビットがセットされている場合、コア・プロセッサはスーパバイザ・ステートにあります。スーパバイザ・ステートではすべての命令が実行可能です。スーパバイザ・ステートで実行される命令によって発生されるバス・サイクルは「スーパバイザ参照」と呼ばれます。コア・プロセッサがスーパバイザ・ステートにある時、スタック・ポインタ(SP)を暗黙に使う命令，またはアドレス・レジスタ(A7)を指定する命令はSSPにアクセスします。例外処理はSビットの状態に無関係に、スーパバイザ・モードで実行されます。例外処理中に発生するバス・サイクルはスーパバイザ参照の部類に入ります。例外処理中のスタック操作はすべてSSPを使います。

5.1.2 ユーザ・ステート

ユーザ・ステートは下位の特権ステートです。命令の実行に対して、ユーザ・ステートはステータス・レジスタのSビットによって決定されます。Sビットがクリアされている場合、コア・プロセッサは命令をユーザ・ステートで実行しています。

ほとんどの命令は、ユーザ・ステートでもスーパバイザ・ステートでも同じ動作をします。しかし、重要なシステム的機能を行ういくつかの命令は特権化(スーパバイザ・モードでしか実行できないように)されています。ユーザ・プログラムではSTOP命令、またはRESET命令の実行は許されません。ユーザ・プログラムからスーパバイザ・モードへの移行は管理された方法でのみ行なわれるようにするため、ステータス・レジスタの上位バイト(システム・バイト)を変える命令は特権化されています。オペレーティング・システムとして使われることになるプログラムのデバッグをやり易くするために、MOVE to USP(USPへの転送命令)，およびMOVE from USP(USPからの転送命令)も特権化されています。

ユーザ・ステートで実行される命令によって発生されるバス・サイクルは「ユーザ・ステート参照」として区別されます。これによって外部のメモリ管理デバイスはアドレスを変換し、アドレス空間のうち保護されている部分へのアクセスを管理することができます。コア・プロセッサがユーザ・ステートで作動している間は、スタック・ポインタ(SP)を暗黙のうちに使う命令，またはアドレス・レジスタ(A7)を指定する命令はUSPにアクセスします。

5.1.3 特権ステートの変更

コア・プロセッサがユーザ・ステートで命令を実行している時、その特権ステートを変更するには例外処理の手段しかありません。例外処理に入ると、その時点でのステータス・レジスタのSビットの値が格納され、Sビットがセットされてコア・プロセッサはスーパバイザ・ステートになります。したがって、例外処理のプログラムはスーパバイザ・ステートで開始されます。

5.1.4 参照の分類

コア・プロセッサが参照を行う時、その種類を3本の機能コード出力信号(FC0~FC2)に符号化して示します。これによって外部でアドレスの変換，アクセスの管理，特殊プロセッサ・ステート(割込みアクノリッジなど)を行うことができます。表5.1に参照の分類を示します。

表5.1 参照の分類

機能コード出力			参照の種類
FC2	FC1	FC0	
L	L	L	*
L	L	H	ユーザ・データ
L	H	L	ユーザ・プログラム
L	H	H	*
H	L	L	*
H	L	H	スーパバイザ・データ
H	H	L	スーパバイザ・プログラム
H	H	H	割込みアクノリッジ

L : LOW　H : HIGH
* : 割当てなし、将来使用予定

5.2 例外処理

割込み，トラップ，およびトレースの詳細を説明する前に、順序として例外処理について一般的に説明します。例外処理は次の4ステップで行われます(例外の原因によって内容が変化する)。

1. ステータス・レジスタの内容が一時的にコピーされ、その後ステータス・レジスタは例外処理用に設定される。
2. 例外ベクタが決定される。
3. 現行のプロセッサ・コンテキストがスーパバイザ・スタックに格納される。
4. 新しい状態を取り込んで、プロセッサは命令の処理に移る。

5.2.1 例外ベクタ

例外ベクタというのは、例外ベクタ・テーブルに格納されている例外を処理するルーチンのメモリ・ロケーションのことです。リセットのベクタ以外の例外ベクタはすべて2ワード(図5.1)です。リセットの例外ベクタは4ワードです。リセット以外の例外ベクタはすべてスーパバイザ・データ空間にあります。リセットの例外ベクタはスーパバイザ・プログラム空間にあります。ベクタ番号を4倍すると例外ベクタのアドレスとなります。ベクタ番号は例外事象の発生原因によって内部的あるいは外部的に発生されます。割込みの場合、割込みアクノリッジのバス・サイクルの間に周辺デバイスが8ビットのベクタ番号(図5.2)をコア・プロセッサのデータ・バス(D0~D7)に乗せます。コア・プロセッサはベクタ番号を32ビットのフル・アドレス(図5.3)に変換します。

例外ベクタのメモリ内でのレイアウトを表5.2に示します。

図5.1 ベクタ・テーブル・エントリのフォーマット

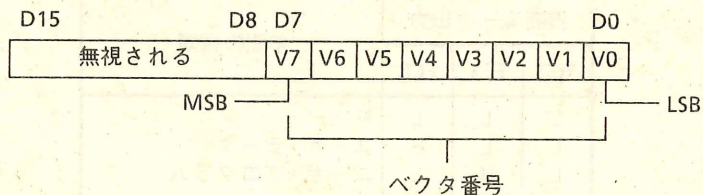



図5.2 ベクタ番号のフォーマット

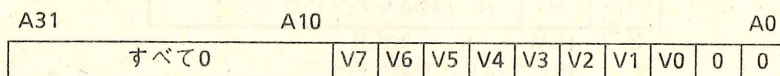



図5.3 ベクタ番号から変換されたアドレス

表5.2が示すように、メモリ内のレイアウトは512ワード(1024バイト)の領域(アドレス0~1023)が使われます。これによって255個のベクタが用意され、そのうちのいくつかは、TRAPおよび他のシステム機能に予約されています。しかし、最初の64個のエントリは保護されていないので、システム設計者によってユーザ割込みベクタをオーバラップさせることができます。

表5.2　例外ベクタの割当て

ベクタ	アドレス			割当て
番号	10進	16進	空間	
0	0	000	SP	リセット:イニシャルSSP
ー	4	004	SP	リセット:イニシャルPC
2	8	008	SD	バス・エラー
3	12	00C	SD	アドレス・エラー
4	16	010	SD	イリーガル命令
5	20	014	SD	0による除算
6	24	018	SD	CHK命令
7	28	01C	SD	TRAPV命令
8	32	020	SD	特権違反
9	36	024	SD	トレース
10	40	028	SD	ライン1010　エミュレータ
11	44	02C	SD	ライン1111　エミュレータ
12*1	48	030	SD	(割当てなし，将来使用予定)
13*1	52	034	SD	(割当てなし，将来使用予定)
14*1	56	038	SD	(割当てなし，将来使用予定)
15	60	03C	SD	初期化されていない割込みのベクタ
16~23*1	64 95	040 05F	SD	(割当てなし，将来使用予定)
24	96	060	SD	にせの割込み
25~31*2	100 124	060 07C	SD	(割当てなし，68000の自動ベクタ)
32~47	128 191	080 0BF	SD	TRAP命令のベクタ
48~63*1	192 255	0C0 0FF	SD	(割当てなし，将来使用予定)
64~255	256 1023	100 3FF	SD	ユーザ割込みベクタ

(注)　SPはスーパバイザ・プログラム空間，SDはスーパバイザ・データ空間を意味します。

*1　:　ベクタ番号12, 13, 14, 16~23, 48~63は将来の機能強化のために予約さています。ユーザの周辺デバイスをこれらの番号に割当てないでください。

*2　:　ベクタ番号25~31 (68000の自動ベクタ) はVPA信号がないため 割当てがありません。

5.2.2 例外事象の種類

例外事象は内部的あるいは外部的に発生することができます。外部的に発生するのは割込み、バス・エラー，およびリセットです。割込みは周辺デバイスからコア・プロセッサの動作を要求するもので、バス・エラーおよびリセットの入力はアクセス管理およびコア・プロセッサの再スタートに使われます。内部的に発生する例外事象としては、命令によるもの，アドレス・エラーによるもの，およびトレース動作によるものがあります。TRAP(トラップ)，TRAPV(オーバフローによるトラップ)，CHK(レジスタの境界値によるチェック)，およびDIV(除算)による各命令は命令実行の時に例外事象を発生する可能性があります。この他、イリーガル命令，奇数アドレスからのワードのフェッチ、および特権違反も例外事象を発生します。トレース動作は優先順位が非常に高い内部割込みとして、命令が実行されるたびに発生します。

5.2.3 例外処理シーケンス

例外処理は識別可能な4つのステップで発生します。最初に、ステータス・レジスタの内容が内部的に保存されます。その後Sビットがセットされ、コア・プロセッサはスーパバイザ・ステートになります。同時にTビットがリセットされ、例外ハンドラ(例外処理ルーチン)の実行がトレース動作によって妨害されないようになります。リセットおよび割込みの場合は割込みマスクも更新されます。

第2のステップでは例外事象のベクタ番号が決定されます。割込みの場合、ベクタ番号はコア・プロセッサがフェッチします。(割込みアクノリッジ・サイクルで)その他の場合はすべて内部回路がベクタ番号を発生します。次にこのベクタ番号を使って例外ベクタのアドレスを発生します。

第3のステップでは、リセット以外は現行のプロセッサ・ステータスを格納します。既知のPC値および前に保存しておいたステータス・レジスタの値がSSPを使ってスタックに格納されます(図5.4)。格納されるPC値は通常は次に実行される命令のアドレスを示しますが、バス・エラーまたはアドレス・エラーの場合は格納されるPC値は予測不可能であり、エラーを発生した命令のアドレスからインクリメントして得られる場合があります。また、現在のプロセッサの内部情報もスタックに格納されます。

最後のステップはすべての例外事象に共通です。新しいPC値が例外ベクタからフェッチされ、コア・プロセッサは命令の実行に戻ります。すなはち、例外ベクタで示されているアドレスから命令フェッチされ、通常の命令デコードおよび実行が開始されます。

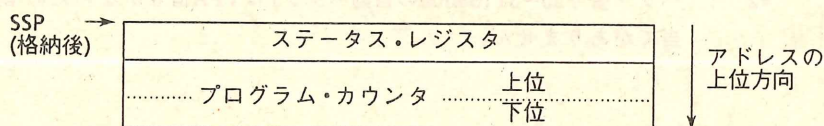



図5.4　例外処理でのスタックへの格納順序
(グループ1および2)

5.2.4 複数の例外事象の同時発生

複数の例外事象が同時に発生した場合の処理について次に説明します。例外事象はその発生原因および優先順位によってグループに分けられます。

グループ0はリセット，バス・エラー，およびアドレス・エラーです。これらの例外事象では、実行中の命令が打ち切られ、例外処理が2クロック・サイクル以内に開始されます。

グループ1はトレース，割込み，特権違反およびイリーガル命令です。これらの例外事象では、実行中の命令は終了しますが、次の命令を実行する前に強制的に例外処理を発生させます。(特権違反およびイリーガル命令は、それらが次に実行される予定の命令である時に検出されます。)

グループ2は命令の通常の処理の一部として発生する例外事象です。TRAP，TRAPV，CHK，および0による除算がこのグループに分類されます。

優先順位はグループ0が最高，グループ2が最低です。グループ0の中ではリセットが最高，次にアドレス・エラー，バス・エラーの順になっています。グループ1の中ではトレースが最高，以下外部割込み，イリーガル命令と特権命令の順になっています。命令は同時には1つしか実行できないので、グループ2の中の優先順位はありません。

2つの例外事象が同時に発生した場合、どちらが先に受け付けられるかは優先順位の関係によって決定されます。したがって、TRAP命令の実行中にバス・エラーが発生した場合、バス・エラーが先に受け付けられ、そのTRAP命令の処理は打ち切られます。別の例では、Tビットがセットしている時に割込みが発生した場合、トレースの方が優先順位が高いので先に処理されます。しかし、命令の処理が再開される前に割込みハンドラ・ルーチンでの命令の処理が開始されます。例外事象のグループおよび優先順位を表5.3に示します。

表5.3 例外事象のグループおよび優先順位

グループ	例外事象	処理
0	リセット アドレス・エラー バス・エラー	例外処理は2クロック・サイクル以内に開始される。
1	トレース 割込み イリーガル命令 特権違反	例外処理は次の命令の前に開始される。
2	TRAP，TRAPV CHK 0による除算	例外処理は命令の通常の実行によって開始される。

5.3 例外事象の詳細説明

例外事象には多くの発生源があり、それぞれ固有の処理が必要です。次に例外の発生源，その発生方法および処理方法について説明します。

5.3.1 リセット

リセットは最高優先レベルの例外事象です。$\overline{\text{RESET}}$がアサートされた時の処理はシステムの初期化、および重大な故障からの回復を目的として設計されています。

$\overline{\text{RESET}}$がアサートされると、その時点で実行されていた処理はすべて打ち切られ、回復することはできません。コア・プロセッサは強制的にスーパバイザ・ステートになり、トレース・ビットはネゲートされます。割込みマスクはレベル7に設定されます。ベクタ番号は内部で自動的に発生されて、スーパバイザのプログラム空間のロケーション0にあるリセット例外ベクタが参照されます。レジスタ(特にスタック・ポインタ)の内容の有効性は保証できないので、PCもSRもセーブされません。リセット例外ベクタの最初の2ワードに入っているアドレスがSSPの最初の値としてフェッチされ、次の2ワードに入っているアドレスが最初のPC値としてフェッチされます。最後に、PCに設定されたアドレスから命令の実行が開始されます。電源投入時/再スタート時のプログラムが最初のPC値によって指定されるようにしておきます。

RESET命令ではリセット・ベクタはロードされないで$\overline{\text{RESET}}$がアサートされて、外部デバイスをリセットします。これにより、ソフトウェアでシステムを既知の状態にリセットし、次の命令で処理を継続することができます。

5.3.2 割込み

割込み要求は、全てインタラプト・コントローラを通してコア・プロセッサへ伝わります。割込み優先度には7つのレベルがあり0から7までの番号が付いており、レベル7が最高位です。ステータス・レジスタには3ビットのマスク・フィールドがあり、その時点での優先順位を示しています。この値以下のレベルの割込みはすべて禁止されます。

割込み要求には外部からの要求と内蔵デバイスからの要求があり、インタラプト・コントローラで各割込み要求ごとに割込みレベルを設定することができます。レベル0は「割込み要求なし」を示します。インタラプト・コントローラでは割込み要求の中から一番優先度の高い要求のレベルでコア・プロセッサに割込みを要求します。コア・プロセッサでは割込み要求をただちに処理するわけではなく、ペンディングにします。ペンディングの割込みは命令と命令の間で検出されます。ペンディングの割込みの優先レベルが現在の割込みレベル以下である場合、コア・プロセッサは次の命令へと進み、その割込みに対する例外処理は後に延期されます。(レベル7の割込みの検出は次に説明するように少し違います。)

ペンディングの割込みの優先レベルが現行割込みレベルより高かった場合、例外処理シーケンスが開始されます。まず、ステータス・レジスタの内容が一時格納され、特権ステートがスーパバイザ・ステートに設定され、トレースが禁止され、コア・プロセッサの優先レベルがアクノリッジされつつある割込みのレベルに設定されます。コア・プロセッサは、インタラプト・コントローラまたは割込みを発生している外部デバイスからベクタ番号をフェッチします。割込みアクノリッジ中にバス・エラーをアサートした場合、その割込みは「にせ」であるとみなされ、にせの割込み用ベクタ番号が発生されます。次にコア・プロセッサは普通の例外処理に進み、PCおよびステータス・レジスタの値をスタックに格納します。格納されるPC値は、割込みが掛からなかった場合に実行される筈であった命令アドレスです。例外ベクタの内容がPCにロードされ、通常の命令実行動作が割込み処理ルーチンにおいて開始されます。割込みアクノリッジのフローチャートを図5.5に,タイミングを図5.6に,割込み処理のタイミング・シーケンスを図5.7に示します。

優先レベル7は特殊ケースです。レベル7の割込みはマスクで禁止できないので、“ノンマスカブル割込み”です。割込み要求レベルがある下位のレベルからレベル7へ変化するたびに割込みが発生します。レベル7の割込みは、要求レベルが7であってコア・プロセッサの優先レベルが命令によって下位のレベルに設定されている場合、レベル比較によっても発生する可能性があることに注意してください。

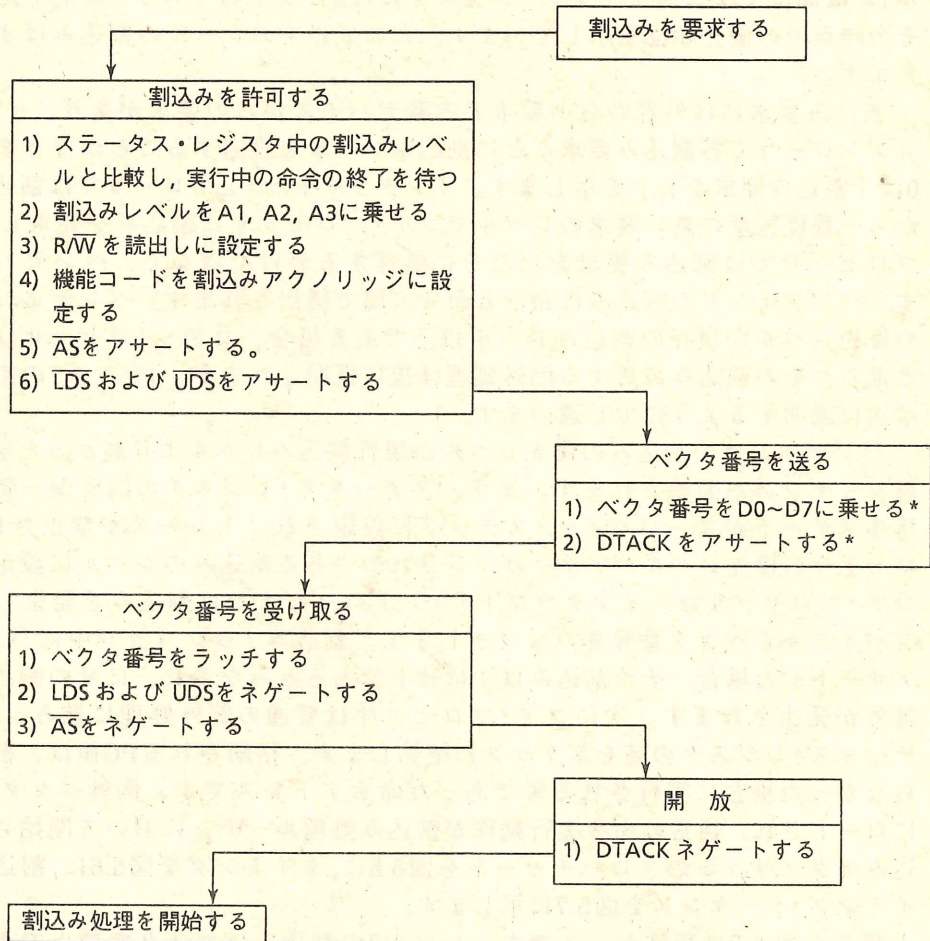



図5.5 割込みアクノリッジ・シーケンスのフローチャート

＊ ： ベクタ番号は1バイトですが、$\overline{\mathrm{UDS}}$, $\overline{\mathrm{LDS}}$の両方がアサートされます。(例外処理のマイクロ・コードがそのようにプログラムされているので。)しかしコア・プロセッサはこの時点でD8~D15を無視します。

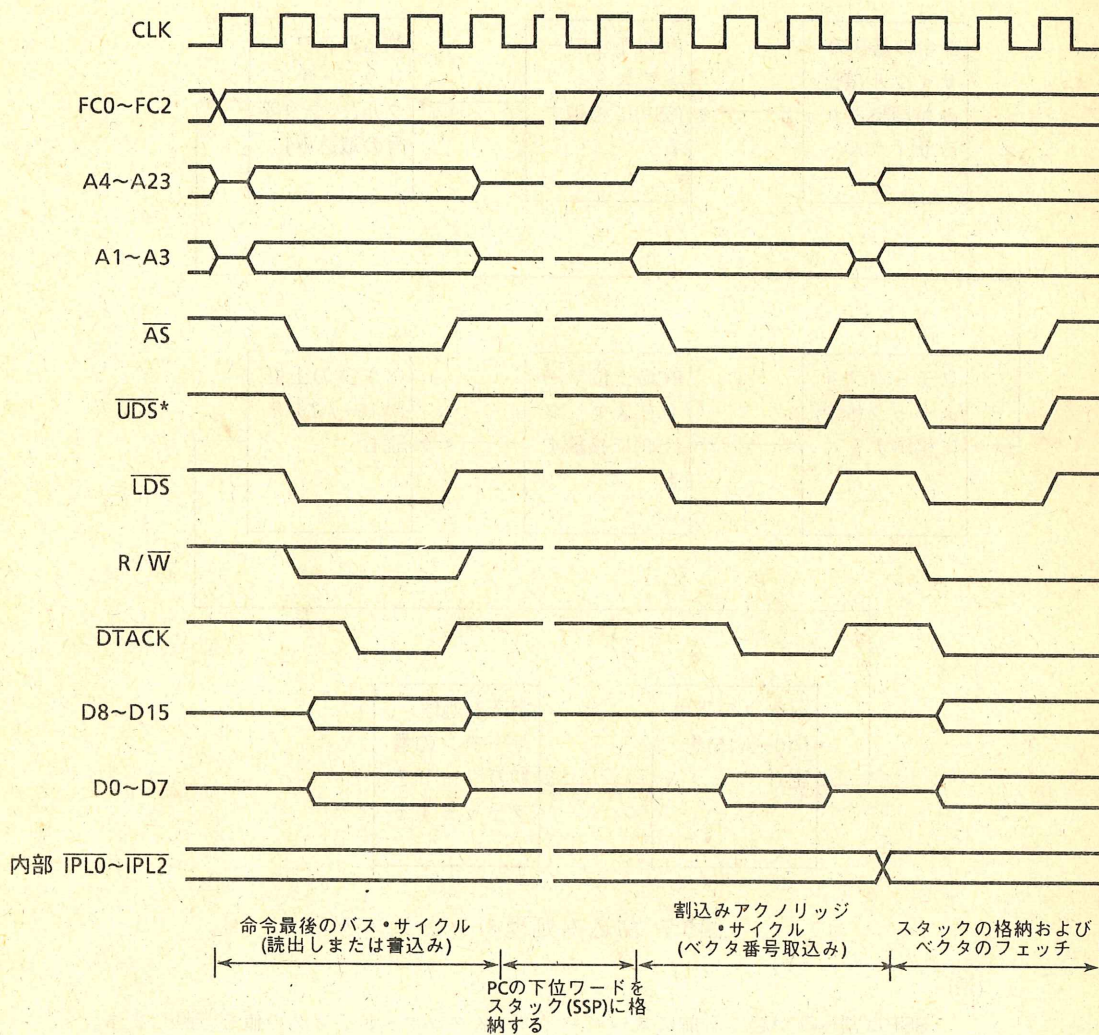



図5.6 割込みアクノリッジ・シーケンスのタイミング

*: ベクタ番号は1バイトですが、$\overline{\text{UDS}}$, $\overline{\text{LDS}}$ の両方がアサートされます。(例外処理のマイクロ・コードがそのようにプリラムされているので。)しかしコア・プロセッサはこの時点でD8~D15を無視します。

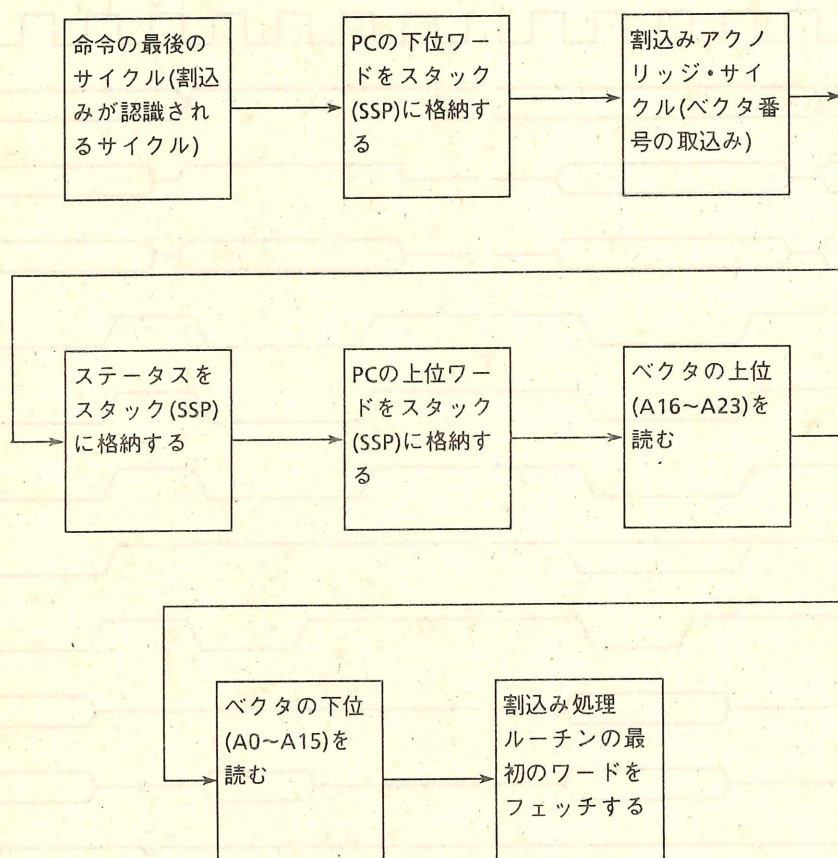



図5.7 割込み処理のシーケンス

(注)

SSPは割込みが起こる前にスーパバイザ・スタック・ポインタの値を参照します。

5.3.3 初期化されていない割込み

割込みを発生しているデバイスは割込みアクノリッジ・サイクル中に割込みベクタをTMP68301に出力します。TLCS-68000ファミリの周辺デバイスは、そのベクタ・レジスタが初期設定されていない場合、ベクタ番号15を出力します。これは一般的にプログラム・エラーからの回復手段として共通的に使われます。

5.3.4 にせの割込み

割込みアクノリッジ・サイクルの間に応答する($\overline{\text{DTACK}}$をアサートする)デバイスがなかった場合、$\overline{\text{BERR}}$をアサートしてベクタの取込みをターミネートする必要があります。この場合、コア・プロセッサはバス・エラー用の例外ベクタの代りににせの割込み用の例外ベクタをフェッチします。

5.3.5 命令トラップ

トラップは命令によって発生される例外事象です。トラップは命令の実行中にプロセッサが異常状態を検出するか、あるいは正常の動作でトラップを起させるような命令を使うことによって発生されます。

わざわざ、トラップを発生させるために使われる命令がいくつかあります。TRAP命令は常に例外事象を強制的に発生させます。この命令はユーザ・プログラムでシステム・コールを実行させるのに便利です。TRAPVおよびCHK命令は、ユーザ・プログラムの実行中にエラー(演算オーバフロー, または配列などの境界オーバー)が検出された場合に例外事象を発生させる命令です。

DIVS(符号付き除算)およびDIVU(符号なし除算)で除数が0であった場合、トラップが発生します。

5.3.6 イリーガル命令および未定義命令

正しい命令の第1ワードのビット・パターンと異なるビット・パターンの命令をイリーガル命令といいます。プログラムの実行中にそのような命令がフェッチされた場合、イリーガル例外事象が発生します。

常にイリーガル命令トラップを発生するビット・パターンは$4AFA, $4AFB, および$4AFCの3種類で、これらはすべてのTLCS-68000ファミリにコンパチブルなマイクロプロセッサに共通です。このうち、$4AFAと$4AFBはシステムに予約されています。$4AFCはユーザ用に確保されています。

ビット15~12が1010または1111であるワードは未定義命令(インプルメントされていない命令)として区別され、別の例外ベクタがこれらのパターンに対して割り当てられているので、効果的なエミュレーションを実行することができます。すなわち、この機能を使ってオペレーティング・システムはプログラム・エラーの検出、またはインプルメントされていない命令のソフトウェアによるエミュレーションを行うことができます。

5.3.7 特権違反

システムを保護する目的で、各種の命令が特権化されています。特権命令の1つをユーザ・ステートで実行しようとすると例外事象が発生します。特権命令には次のものがあります。

STOP	ANDイミディエート to SR
RESET	EOR イミディエート to SR
RTE	OR イミディエート to SR
MOVE to SR	MOVE USP

5.3.8 トレース動作

プログラム開発を支援するために、TMP68301には命令が1つ実行されるたびにトレースできる機能があります。トレース・ステートでは、命令が1つ実行されるたびに強制的に例外事象が発生されるので、デバッグ・プログラムにより、テストされるプログラムの実行をモニタすることができます。

トレース機能は、ステータス・レジスタのスーパバイザ部分にあるTビットを使います。Tビットがリセットされている時はトレースが禁止されており、命令は普通に次々に実行されます。命令の実行開始時点でTビットがセットされている場合、その命令の実行終了時点でトレース例外事象が発生します。割込みが発生したか、あるいはイリーガルまたは特権命令であったためにその命令が実行されなかった場合は、トレース例外事象は発生しません。リセット，バス・エラー，またはアドレス例外事象によって命令が打ち切られた場合もトレース例外事象は発生しません。命令が実際に実行され、その終了時点で割込みがペンディングになっていた場合、トレース例外処理が割込み処理より前に実行されます。命令の実行中に、その命令自身による例外事象が発生した場合は、その例外事象の方がトレース例外事象より前に処理されます。

極端な例として、トレースが許可されている状態でTRAP命令が実行されている間に割込み要求が入力された場合を考えます。この場合、まず，トラップ例外処理が行われ、次にトレース例外処理，最後に割込み例外処理が実行されます。命令の実行は割込み処理ルーチンにおいて再開されます。

5.3.9 バス・エラー

バス・エラー例外事象は外部回路がバス・エラーを例外事象として処理するよう要求してきた時に発生し、実行中のバス・サイクルは打切られます。プロセッサが命令を実行していた場合でも例外処理を実行した場合でも、その動作は打切られ、プロセッサはただちに例外処理を開始します。

バス・エラーの例外処理は通常のシーケンスのステップに従って実行されます。すなわち、ステータス・レジスタの内容が一時保存され、スーパバイザ・ステートに入り、トレース・ステートはオフされます。ベクタ番号を発生し、バス・エラーのベクタを参照します。

バス・エラーは命令の境界で発生するとは限らないので、普通の例外事象の場合より更に詳しい情報がスーパバイザ・スタックに格納されます。PCおよびステータス・レジスタの内容は勿論格納されます。格納されるPC値はバス・エラーを発生した命令の最初のワードのアドレスからいくらか(2~10バイト)進んだアドレスとなります。次の命令をフェッチしている間にバス・エラーが発生した場合(その命令が分岐,ジャンプまたはリターン命令であっても)退避されたPC値は、現在の命令の付近の値を示していることがあります。通常の情報の他にプロセッサは処理中の命令の第1ワードの内部コピー, および打切られたバス・サイクルによってアクセスされていたアドレスを格納します。すなわち、読出しであったか書込みであったか、コア・プロセッサが命令を処理していたかどうか, およびバス・エラー発生時に出力されていた機能コードが格納されます。コア・プロセッサがノーマル ステートにある時またはグループ2例外処理を実行している時が命令を処理している状態で、グループ0またはグループ1の例外事象を処理している時が命令を処理していない時です。この情報がスーパバイザ・スタックに格納されている様子を図5.8に示します。

	15 … 5	4	3	2 1 0
SSP → (格納後)		R/$\overline{W}$	I/N	機能コード
	アクセス・アドレス 上位			
	アクセス・アドレス 下位			
	命令レジスタ			
	ステータス・レジスタ			
	プログラム・カウンタ 上位			
	プログラム・カウンタ 下位			

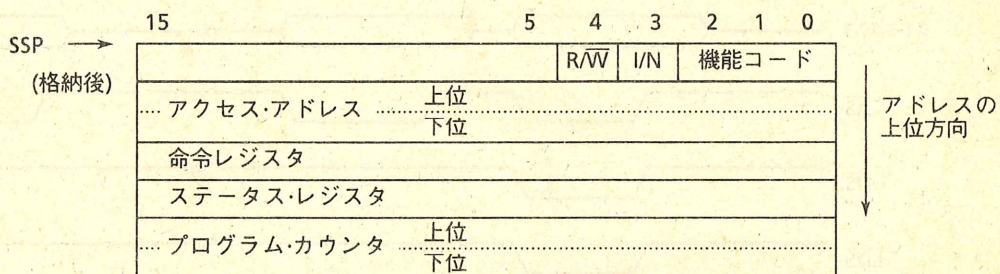
↓ アドレスの上位方向

R/$\overline{W}$ (読出し/書込み) : 書込み = 0, 読出し = 1
I/N (命令/命令でない) : 命令 = 0, 命令でない = 1

図5.8 スーパバイザ・スタックの格納順序(グループ0)
(リセットの場合は除く)

この情報だけではバス・エラーから完全に回復するには不十分ですが、ソフトウェアの診断には有効です。最後に、コア・プロセッサはベクタ番号2の中に含まれているアドレスから命令の処理を開始します。スタックをもとに戻して、どこから継続するかを決めるのはバス・エラー処理ルーチンの責任です。

バス・エラー, アドレス・エラー, またはリセットに対する例外処理が行われている間にバス・エラーが発生した場合、コア・プロセッサは停止し、すべての処理は止まります。これによってシステムのプログラムを破壊する様な故障を事前に検出することができます。というのは、コア・プロセッサがシステムから切り離されるので、メモリ内容はすべて破壊してしまうようなことにはならないからです。停止中のコア・プロセッサを再スタートさせる方法は$\overline{\text{RESET}}$をアサートするしかありません。

5.3.10 アドレス・エラー

アドレス・エラーはコア・プロセッサがワードまたはロング・ワードのオペランド，または命令を奇数アドレスでアクセスしようとした時に発生します。これは内部的に発生したバス・エラーとよく似ている、そのバス サイクルは打切られ、コア・プロセッサは実行中の処理をすべて中止し、例外処理を開始します。例外処理が開始された後のシーケンスは格納される情報を含めてバス・エラーの場合と同じです。ただし、ベクタ番号はアドレス・エラーに対するものとなります。バス・エラーの場合と同様に、バス エラー，アドレス・エラー，またはリセットの例外処理中にアドレス・エラーが発生した場合、コア・プロセッサは停止します。図5.9に示すように、アドレス・エラーの場合は短いバス サイクルが1つあって、その後に例外処理が続きます。

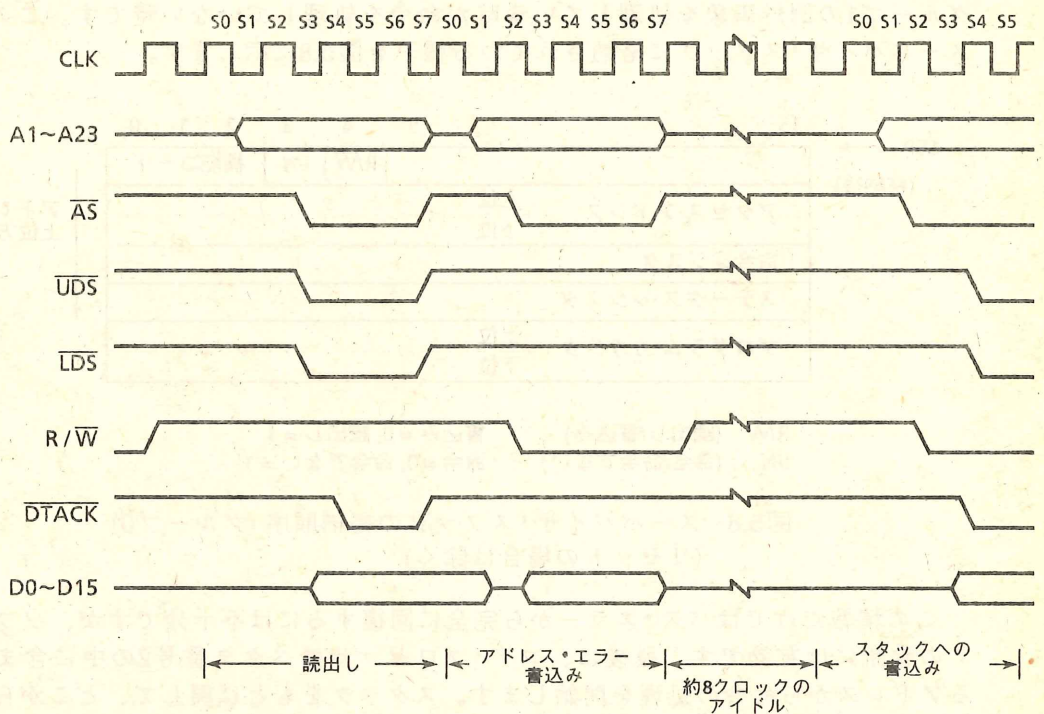



図5.9 アドレス・エラーのタイミング

第6章　周辺機能ブロック

6.1　アドレス・デコーダ

6.1.1 概要

アドレス・デコーダでは2つの$\overline{\text{CS}}$(Chip Select)に対応したメモリ(又は外部デバイス)用のエリア(以下メモリ・エリア)と内蔵デバイスのレジスタ用エリア(以下レジスタ・エリア)を選択することができます。各エリアはコア・プロセッサが持つ16Mバイトのアドレス空間の中に自由に配置することができます。メモリ・エリアはエリアの大きさも変えることができます。さらに、各メモリ・エリアごとに0~7クロックのウェイト・サイクルを挿入するよう$\overline{\text{DTACK}}$信号を発生することができます。

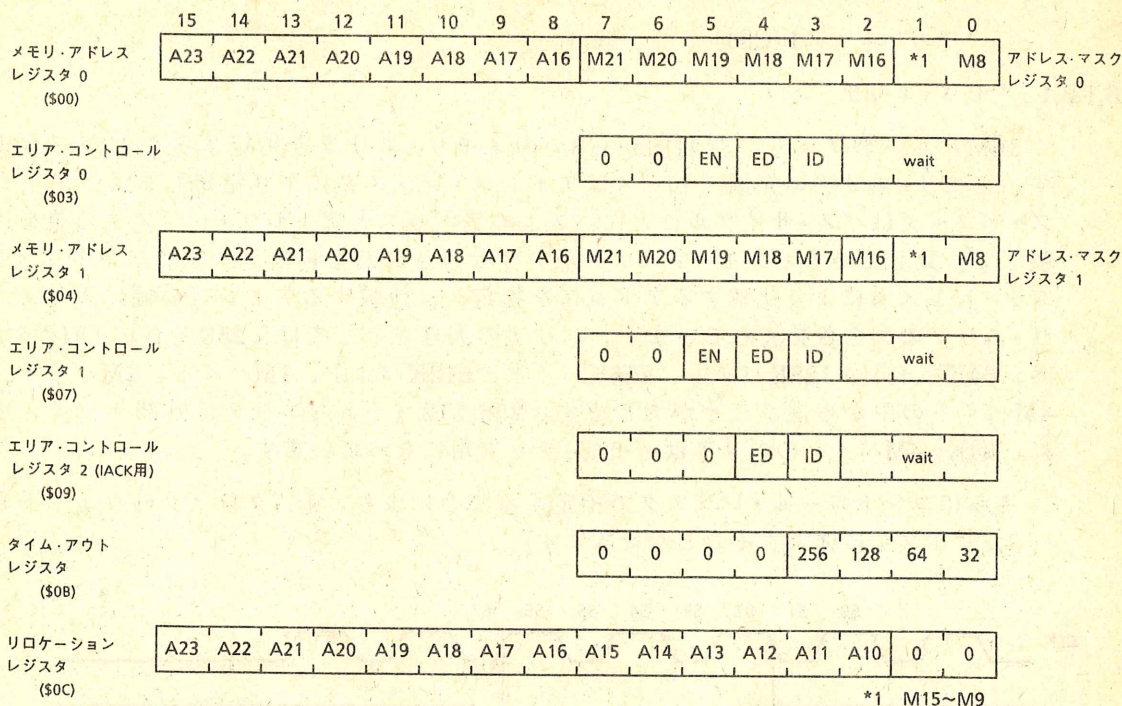
また、バスサイクルのタイムアウトを検出して$\overline{\text{BERR}}$(Bus Error)を発生する機能も持っています。

(注)　(　)内はアドレス・オフセットであり、これにリロケーションレジスタの値を加えた値が、実アドレスになります。

図6.1 アドレス・デコーダ・レジスタ・マップ

6.1.2 エリアの選択

アドレス・デコーダでは各バス・サイクルごとにアドレスをデコードして2つのメモリ・エリア、又はレジスタ・エリアがアクセスされているかをチェックしています。

もしメモリ・エリアがアクセスされていると外部にチップ・セレクト信号($\overline{\text{CSn}}$)を

アサートし、あらかじめ設定されたウェイト数で$\overline{\mathrm{DTACK}}$をコア・プロセッサに送ります。もしレジスタ・エリアがアクセスされると内蔵されているデバイスを選択してデータの転送を行います。この時、外部の端子には他のエリアをアクセスした時と同様に有効な信号が出力されます。ただしリード・サイクルの時、外部からのデータは無視され内蔵デバイスからのデータが読込まれます。

各エリアの先頭アドレスはアドレス・レジスタ(又はリロケーション・レジスタ)で指定します。もし各エリアを同じアドレス空間に設定した場合下記の優先順位に従いエリアを選択します。(割込みのIACKサイクルでは、どちらのエリアも選択されません。)

エリア	優先順位
レジスタ・エリア	高
メモリ・エリア0(CS0)	｜
メモリ・エリア1(CS1)	低

6.1.2.1 メモリ・エリア

16Mバイトのアドレス空間内から2つのメモリ・エリアを指定することができます。メモリ・エリアの先頭アドレスはアドレス・レジスタにより指定します。アドレス・デコーダはバス・サイクルごとにバス上のアドレスとアドレス・レジスタの値を比較して一致しているとメモリ・エリアがアクセスされたと判断します。アドレス・マスク・レジスタにより比較するアドレスを指定し、比較するアドレスの幅によりメモリ・エリアの大きさを決めています。エリアの大きさとしては、256バイト、512バイト、64Kバイト、128Kバイト、256Kバイト、512Kバイト、1Mバイト、2Mバイト、4Mバイトの中から選ぶことができます。256, 512バイトのエリアは外部デバイス用に、64K~4Mバイトのエリアはメモリ・チップ用になっています。

さらにコントロール・レジスタで指定することにより、0~7クロックのウェイトをバス・サイクルに挿入することができます。

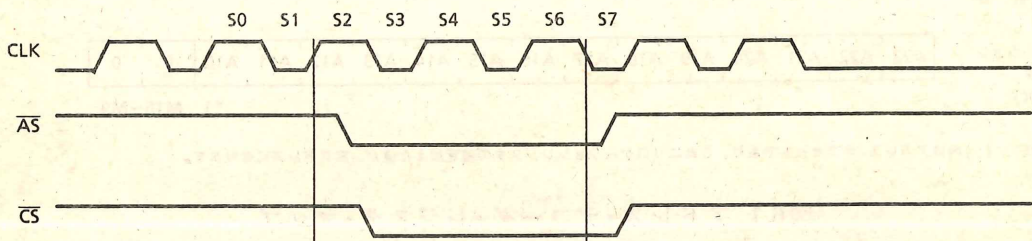



図6.2 CS信号とAS信号の関係

6.1.2.2 レジスタ・エリア

レジスタ・エリアにはアドレス・デコーダも含め内蔵デバイスのレジスタが集められています。レジスタ・エリアの大きさは1Kバイトで、先頭アドレスをリロケーション・レジスタで指定することにより1Kバイト境界単位でアドレス空間の任意の位置に置くことができます。各内蔵デバイスのレジスタは下の図のようになっています。リセット後は$FFFC00(アドレス空間の後)に置かれます。

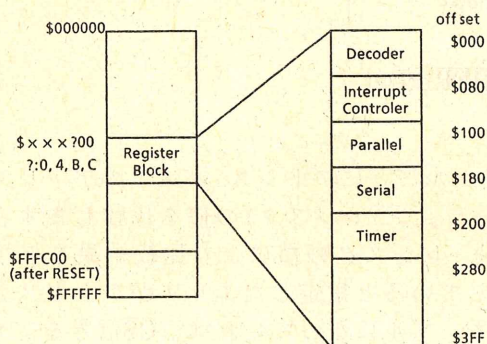



図6.3 レジスタ・ブロック・マップ

6.1.2.3 DTACKの自動発生 (wait数可変)

アドレス・デコーダでは各エリアごとに$\overline{\mathrm{DTACK}}$信号を発生させることが可能です。レジスタ・エリアに対しては0ウェイトの$\overline{\mathrm{DTACK}}$を自動的に発生します。メモリ・エリアに対しては外部の回路で作った$\overline{\mathrm{DTACK}}$とアドレス・デコーダで作った$\overline{\mathrm{DTACK}}$のどちらか又は両方(早くアサートされた方)を選択して使うことができます。アドレス・デコーダで作る$\overline{\mathrm{DTACK}}$には0~7クロックのウェイトを挿入することができます。さらに、IACKサイクル(外部からベクタ・ナンバーを読む場合)でもメモリ・エリアと同様な機能があります。

6.1.3 バス・サイクルの監視

メモリの実装されていないアドレス空間をアクセスした時のように、$\overline{\mathrm{DTACK}}$が返ってこないバス・サイクルにそなえて、バス・サイクルの長さを監視して異常があれば$\overline{\mathrm{BERR}}$信号を発生する機能があります。バス・エラーまでのバス・サイクルの長さは、32, 64, 128, 256クロックの中から選ぶことができます。

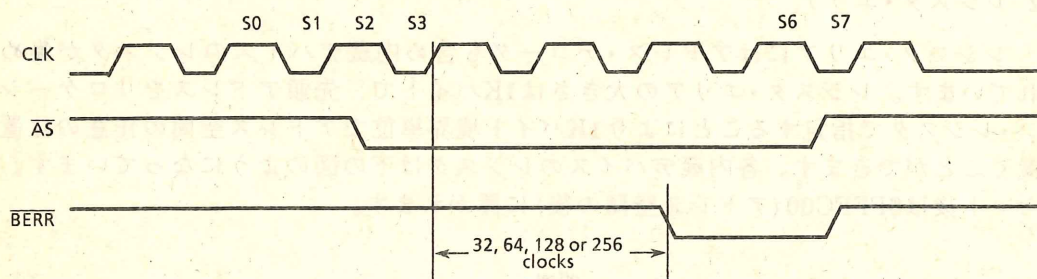



図6.4 $\overline{\text{BERR}}$信号

6.1.4 エリアの選択方法

アドレス・デコーダでは各バス・サイクルごとにアドレス・バス上のアドレスと各メモリ・アドレス・レジスタ(及びリロケーション・レジスタ)の値を比較します。このアドレスを比較する時にアドレス・マスク・レジスタの値により比較結果を無視させることができます。比較した結果が一致していると指定したエリアがアクセスされたと見なします。そして、このエリアがイネーブルになっていれば、$\overline{\text{CS}}$信号をアサートし指定されていれば$\overline{\text{DTACK}}$を発生します。もし複数のエリアが同じアドレス空間に割当てられた場合は前記(6.1.2)の優先順位に従いイネーブルになっているエリアの中で最も優先順位の高いエリアが有効となります。

6.1.5 レジスタ構成

6.1.5.1 メモリ・アドレス・レジスタ

メモリ・エリアの先頭アドレスを指定します。指定できるアドレスはA16~A23の8ビットです。メモリ・エリアの先頭はメモリ・エリアの大きさの境界ごとにしか設定できません(最小64Kバイト、最大4Mバイト。エリアの大きさが256,512バイトの時は64Kバイト境界になります)。

リセット後はメモリ・アドレス・レジスタ0(CS0用)が$00になります。メモリ・アドレス・レジスタ1は不定です。

15	14	13	12	11	10	9	8
A23	A22	A21	A20	A19	A18	A17	A16

(オフセット $00 CS0
$04 CS1)

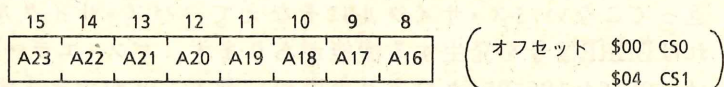



図6.5 メモリ・アドレス・レジスタ

6.1.5.2 アドレス・マスク・レジスタ

A8~A23の中から比較するアドレスを指定します。比較するアドレスの幅によりメモリ・エリアの大きさが変ります。各ビットごとにマスクするアドレスは下記のようになっています。

```
bit    mask address
0      A8
1      A9-A15
2      A16
3      A17
4      A18
5      A19
6      A20
7      A21
```

各ビットとも1でマスクされ、0でマスクされません。

リセット後はアドレス・マスク・レジスタ0(CS0用)が$FF(メモリ・エリアの大きさが4Mバイト)になります。

アドレス・マスク・レジスタ1は不定です。

7	6	5	4	3	2	1	0
M21	M20	M19	M18	M17	M16	*1	M8

*1 M15-M9

(オフセット $01 CS0
$05 CS1)

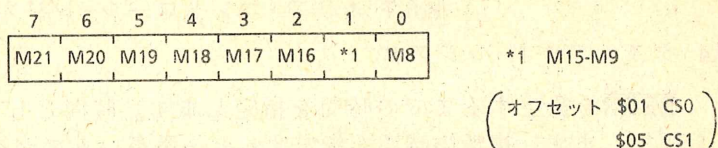



図6.6 アドレス・マスク・レジスタ

6.1.5.3 エリア・コントロール・レジスタ

エリアのイネーブル(1)/ディセーブル(0)、DTACKのモード、挿入するウェイト数を設定します。エリアをディセーブルにするとエリアがアクセスされてもCS信号をアサートせず、アドレス・デコーダからの$\overline{\mathrm{DTACK}}$発生もしません。

bit 0~2 ウェイト数を指定します。

bit 3 アドレス・デコーダの$\overline{\mathrm{DTACK}}$の
イネーブル(1)/ディセーブル(0)を指定します。

bit 4 外部の$\overline{\mathrm{DTACK}}$の
イネーブル(1)/ディセーブル(0)を指定します。

bit 5 エリアのイネーブル(1)/ディセーブル(0)を指定します。

DTACKモードは外部からの$\overline{\mathrm{DTACK}}$とアドレス・デコーダからの$\overline{\mathrm{DTACK}}$のどちらを使うのかを決めるもので下記の様になっています。

bit4, 3	mode
11	両方使う
10	外部の$\overline{\mathrm{DTACK}}$を使う
01	アドレス・デコーダの$\overline{\mathrm{DTACK}}$を使う
00	指定不可(指定しても"01"になります)

リセット後はエリア・コントロール・レジスタ0(CS0用)が$3D(エリアはイネーブル、外部/内部DTACKはイネーブル、ウェイト5クロック)になり、エリア・コントロール・レジスタ1は$18(エリアはディセーブル、外部/内部DTACKはイネーブル、ウェイト0クロック)になります。

エリア・コントロール・レジスタ2は、外部割込みにおける割込みアクノリッジサイクルのDTACK用で、リセット後は$18(エリアは常時イネーブル、外部/内部 DTACKはイネーブル、ウェイト0クロック)になります。内蔵周辺からの割込みはウエイト0クロック固定です。

7	6	5	4	3	2	1	0
0	0	EN *1	ED	ID	wait		

*1 エリアコントロールレジスタ2ではこのbitはありません。

(オフセット $03 CS0
$07 CS1
$09 IACK)

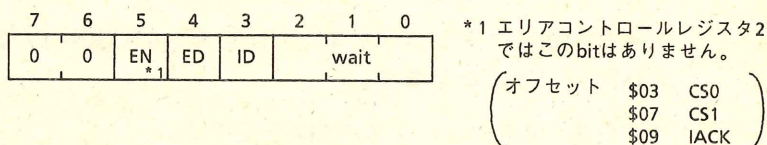



図6.7 エリア・コントロール・レジスタ

6.1.5.4 タイムアウト・レジスタ

$\overline{\text{BERR}}$を発生するまでの時間を指定します。時間としては、32, 64, 128, 256の中から選択します。複数の時間を指定すると一番長いものが選ばれ、他のビットはクリアされます。どれも選ばなかった場合は、$\overline{\text{BERR}}$を発生しません。

bit 0　32クロック後を指定します。(1:on、0:off)

bit 1　64クロック後を指定します。(1:on、0:off)

bit 2　128クロック後を指定します。(1:on、0:off)

bit 3　256クロック後を指定します。(1:on、0:off)

リセット後は$08(256クロック)になっています。

7	6	5	4	3	2	1	0
0	0	0	0	256	128	64	32

(オフセット $0B)

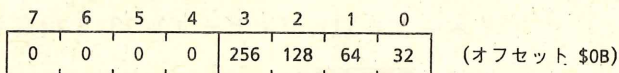



図6.8 タイムアウト・レジスタ

6.1.5.5 リロケーション・レジスタ

レジスタ・ブロックの先頭アドレスを指定します。指定するアドレスはA10~A23です。従ってレジスタ・ブロックの先頭は1Kバイトの境界にしか設定できません。

リセット後は$FFFCになっています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	0	0

(オフセット $0C)

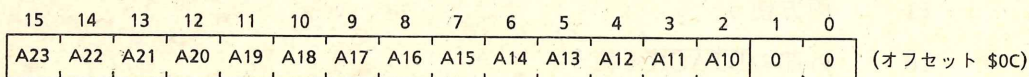



図6.9 リロケーション・レジスタ

6.1.6 外部バス・マスタによるバス・サイクル

コア・プロセッサがバス制御権を外部のバス・マスタに渡し、外部バス・マスタがバス・サイクルを発生する場合も、アドレス・デコーダはコア・プロセッサがバス・マスタであるのと同じように働きます。その結果、外部マスタからも内部レジスタをアクセスすることでができます。この時、外部バス・マスタによるバス・サイクルのタイミングは、コア・プロセッサの発生するバス・サイクルの仕様を満足していなければなりません。そうでないと、例えば、外部バス・マスタによるメモリ書込みサイクルで、内部レジスタの内容が書き変わってしまう危険があります。

6.2 インタラプト・コントローラ

6.2.1 概要

このインタラプト・コントローラは10個の割込みチャネルを持っています。そのなかの、7チャネルを内蔵デバイスに、3チャネルを外部からの要求に割り当てています。割込みレベルは各チャネルごとに設定でき、優先順位を自由に変えられます。インタラプト・コントローラではIACKサイクルに合せて対応するチャネルごとにIACK信号を発生します。ベクタナンバ発生は内部で自動的におこなうことができますが、外部割込みの場合、ベクタ・ナンバーを外から入れることもできます。ベクタナンバ自動発生により、AUTOベクタ割込みの代わりに使うことができます。

(offset)

*1

T2 : Timer ch2
T1 : Timer ch1
T0 : Timer ch0
P　: Parallel
S2 : Serial ch2
S1 : Serial ch1
S0 : Serial ch0
E2 : External ch2
E1 : External ch1
E0 : External ch0

図6.10 インタラプト・コントローラ・レジスタ・マップ

6.2.2 割込み要求

インタラプト・コントローラではチャネルに割込み要求があると、あらかじめ設定されているレベルでIPL0~2を使ってコア・プロセッサに割込み要求をかけます。いくつかのチャネルで同時に要求が発生した場合は優先順位の一番高い要求を出します。

6.2.2.1 要求の入力モード

外部からの要求の場合、入力モードとしては下記の4つのモードがあります。

ロー・レベル

ハイ・レベル

立上りエッジ

立下りエッジ

レベル・モードの場合は割込み要求をIACKが返ってくるまで出し続けなければなりません。

(外部からの要求信号はクロックの立下がりでサンプリングされます。)

(エッジの場合、エッジの後同じ状態が2クロック以上必要です。)

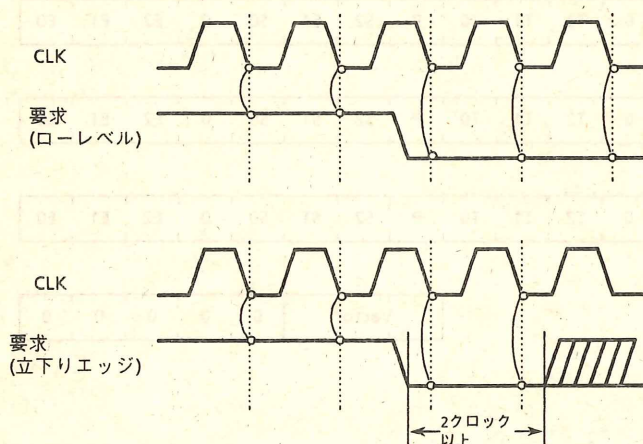



図6.11 割込み要求

6.2.3 チャネル間の優先順位

6.2.3.1 チャネルの割込み要求のレベル

インタラプト・コントローラでは各割込みチャネルごとに割込み優先順位を設定できます。優先順位は設定したレベルによってきまります。(高7~1低)

6.2.3.2 同じレベルのチャンネル間の優先順位

もし複数のチャンネルが同じレベルに設定されていた場合、下記の優先順位に従います。

```
high  external 0
 |    timer 0
 |    serial 0
 |    parallel
 |    external 1
 |    timer 1
 |    serial 1
 |    serial 2
 |    timer 2
low   external 2
```

6.2.4 IACKサイクル

6.2.4.1 IACK信号

コアプロセッサの発生したIACKサイクル時、インタラプト・コントローラではA1~3をデコードして受け付けられた割込みレベルを検出して、対応するチャンネルのIACK信号を発生します。もし、いくつかのチャネルが同じレベルで要求を出していた場合は、上記(6.2.3.2)の優先順位に従って最も優先順位の高いチャネルに対するIACK信号を発生します。(IACK信号はASと同じタイミングでアサートされます。)

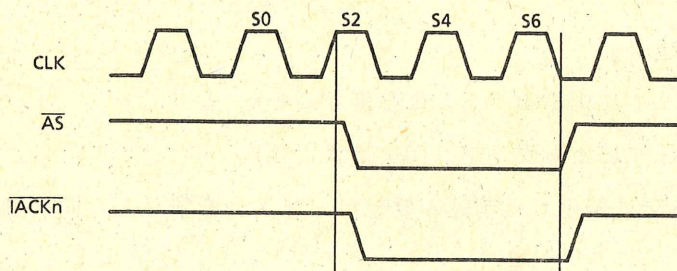



図6.12 IACK信号

6.2.4.2 ベクタ・ナンバー

インタラプト・コントローラはIACKサイクルに合わせてベクタ・ナンバーを発生します(オートベクタ割込みの代わりになります)。各チャンネルのベクタ・ナンバーの内、上位3ビットはベクタ・ナンバー・レジスタで指定し、下位5ビットは次のように各チャンネルごと決まっています。

ch	vector NO.	
external 0	00000	
external 1	00001	
external 2	00010	
timer 0	00100	
timer 1	00101	
timer 2	00110	
serial 0	01000	error, break
serial 0	01001	buffer full
serial 0	01010	buffer empty
serial 0	01011	シリアル割込ペンディング中に割込み元がクリアされた
serial 1	01100	error, break
serial 1	01101	buffer full
serial 1	01110	buffer empty
serial 1	01111	シリアル割込ペンディング中に割込み元がクリアされた
serial 2	10000	error, break
serial 2	10001	buffer full
serial 2	10010	buffer empty
serial 2	10011	シリアル割込ペンディング中に割込み元がクリアされた
parallel	10100	受信終了
parallel	10101	転送可能
parallel	10110	外部機器の状態変化
parallel	10111	パラレル割込みペンディング中に割込み元がクリアされた

外部からの割込みの場合、ベクタ・ナンバーはIACKサイクルで外部から入力することもできます。

6.2.5 割込み状態

各チャネルの状態は、マスク、ペンディング、インサービスの各レジスタに示めされます。以下の説明で、set:以下はそのビットをセットする手段を示し、reset:以下はリセットする手段を示します。

6.2.5.1 マスク・ビット(M)

割込み要求を許可します。

1: 要求を無視します。(マスクする)

0: 要求を受け付けます。(マスクしない)

set　:　リセット信号。

　　　　ソフトウェアによるセット。(1を書きこむ)

reset:　ソフトウェアによるクリア。(0を書きこむ)

マスクされている時に発生した割込み要求は、マスクを解除した時に受け付けられます。

割込みレベルは、マスクされている時のみ書換え可能です。

6.2.5.2　ペンディング・ビット(P)

割込み要求が発生していることを示します。

1: 割込み要求があることを示します。

0: 割込み要求の無いことを示します。

set　:　割込み要求が発生した。

reset:　割込み要求が受け付けられた。

　　　　ソフトウェアによるクリア。(0を書きこむ)

　　　　リセット信号。

このビットはソフトウェアではセットできません。

外部割込みがレベル・モードの時、ソフトウェアによりこのビットをクリアするには、あらかじめ割込み要求を止めておかなければなりません。

6.2.5.3　インサービス・ビット(I)

割込み要求が受け付けられたことせ示します。

1: 割込み要求を受け付けたことを示します。

0: 割込み要求を受け付けていないことを示します。

set　:　割込み要求を受け付けた。

reset:　ソフトウェアによるクリア。(0を書きこむ)

　　　　リセット信号。

このビットはソフトウェアではセットできません。

マスク・ビット(M)、ペンディング・ビット(P)、インサービス・ビット(I)の値による割込みの状態

M	P	I	
0	0	0	要求がない

0	0	1	割込み処理ルーチン中
0	1	0	割込み要求が来た
0	1	1	割込み処理ルーチン中に次の割込み要求が来た
1	0	0	要求がない
1	0	1	割込み処理ルーチン中でマスクされた
1	1	0	マスクされているところに割込み要求が来た
1	1	1	割込み処理ルーチン中でマスクしたところに割込み要求が来た

6.2.6 レジスタ構成

6.2.6.1 コントロール・レジスタ

6.2.6.1.1 インタラプト・コントロールレジスタ0~2

外部割込みに対応するレジスタです。

bit2~0は割込みレベルを指定。

bit3, 4は要求モードの指定であり、下記のようになります。

bit4 R/F	bit3 L/E	mode
0	0	立下りエッジ
1	0	立上りエッジ
0	1	ロー・レベル
1	1	ハイ・レベル

bit5はベクタナンバ自動発生機能のON (1) /OFF (0) です。ONの時、内部で自動的にベクタナンバを発生します。(6.2.4.2 ベクタ・ナンバー参照)

リセット後の初期値は$07 (立下りエッジ、割込みレベル7) になっています。

このレジスタは、マスク・レジスタで割込みがマスクされている時のみ書換え可能です。

図6.13 インタラプト・コントロール・レジスタ

6.2.6.1.2 インタラプト・コントロール・レジスタ3~9

内蔵デバイスに対応するレジスタであり、各内蔵デバイスが発生する要求のレベルを指定します(bit0~2)。対応する内蔵デバイスは下記のようになります。

REGISTER	DEVICE
3	Serial ch 0
4	Serial ch 1
5	Serial ch 2
6	Parallel
7	Timer ch 0
8	Timer ch 1
9	Timer ch 2

リセット後の初期値は$07(割込みレベル7)になっています。マスクレジスタで割込みがマスクされている時のみ書き換え可能です。

図6.14インタラプト·コントロール·レジスタ

6.2.6.2 マスク・レジスタ

各チャネルのマスクを設定します。"1"でマスク(割込みを無視する)、"0"でマスクしない状態になります。マスクを途中でかける場合は変更するビットの割込みが生じても受けつけない状態(例:割込みレベル7)でおこなって下さい。マスクをかけると同時に割込み要求が入り、コアCPUが受けつけると誤った割込みベクタを発生することがあります。

リセット後の初期値は$07F7(全チャネルマスク)になっています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	T2	T1	T0	P	S2	S1	S0	0	E2	E1	E0

*1 (オフセット $94)

図6.15 インタラプト・マスク・レジスタ

6.2.6.3 ペンディング・レジスタ

割込み要求があること、およびその割込み要求がまだコア・プロセッサに受け付けられていないことを示します。"1"でコア・プロセッサに受け付けられていない割込み要求があることを、"0"で割込み要求がないことを示します。

各ビットはコア・プロセッサに要求が受け付けられると自動的にクリアされます。プログラムによりクリアすると割込み要求をキャンセルしたことになります。

リセット後の初期値は$0000になっています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	T2	T1	T0	P	S2	S1	S0	0	E2	E1	E0

*1 (オフセット $96)

図6.16 インタラプト・ペンディング・レジスタ

6.2.6.4 インサービス・レジスタ

割込み要求がコア・プロセッサに受け付けられたことを示します。“1”で要求が受け付けられたことを、“0”で要求がまだ受け付けられていないことを示します。

注)“1”が立っていても動作には影響しませんが、各ビットは割込みハンドラー・ルーチンのなかでクリアして下さい。(自動的にはクリアしません)

リセット後の初期値は$0000になっています。

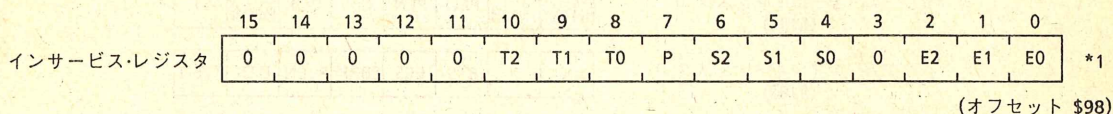



図6.17 インタラプト・インサービス・レジスタ

6.2.6.5 ベクタ・ナンバー・レジスタ

ベクタ・ナンバーの上位3ビットを指定します。ベクタ・ナンバーの下位5ビットは割込みチャネルごとに決っています。

bit7~5でベクタ・ナンバーの上位3ビットを指定します。他のビットは読出すと“0”になります。リセット後の初期値は$00になっています。

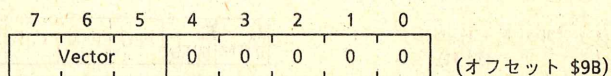



図6.18 ベクタ・ナンバー・レジスタ

*1 T2:Timer ch 2
T1:Timer ch 1
T0:Timer ch 0
P :Parallel
S2:Serial ch 2
S1:Serial ch 1
S0:Serial ch 0
E2:External ch 2
E1:External ch 1
E0:External ch 0

6.3 シリアル・インターフェース

6.3.1 概要

このシリアル・インターフェースは非同期通信をサポートする完全に独立した3チャネルで構成されています。さらにチャネルごとに割込み要求を発生する事ができます。

レジスタ	7	6	5	4	3	2	1	0
シリアル・モード・レジスタ 0 ($181)	RX INTM	ER INTM	PEO	PEN	CL1	CL0	TX INTM	ST
シリアル・コマンド・レジスタ 0 ($183)	X	X	RTS	ERS	SBRK	RXEN	DTR	TXEN
シリアル・ボーレート・レジスタ 0 ($185)	B7	B6	B5	B4	B3	B2	B1	B0
シリアル・ステータス・レジスタ 0 ($187)	DSR	PBRK	FE	OE	PE	TXE	RX RDY	TX RDY
シリアル・データ・レジスタ 0 ($189)	D7	D6	D5	D4	D3	D2	D1	D0
シリアル・プリスケラ・レジスタ ($18D)	P7	P6	P5	P4	P3	P2	P1	P0
シリアル・コントロール・レジスタ ($18F)	CKSE	X	RES	X	X	X	X	INTM
シリアル・モード・レジスタ 1 ($191)	RX INTM	ER INTM	PEO	PEN	CL1	CL0	TX INTM	ST
シリアル・コマンド・レジスタ 1 ($193)	X	X	RTS	ERS	SBRK	RXEN	DTR	TXEN
シリアル・ボーレート・レジスタ 1 ($195)	B7	B6	B5	B4	B3	B2	B1	B0
シリアル・ステータス・レジスタ 1 ($197)	DSR	PBRK	FE	OE	PE	TXE	RX RDY	TX RDY
シリアル・データ・レジスタ 1 ($199)	D7	D6	D5	D4	D3	D2	D1	D0
シリアル・モード・レジスタ 2 ($1A1)	RX INTM	ER INTM	PEO	PEN	CL1	CL0	TX INTM	ST
シリアル・コマンド・レジスタ 2 ($1A3)	X	X	RTS	ERS	SBRK	RXEN	DTR	TXEN
シリアル・ボーレート・レジスタ 2 ($1A5)	B7	B6	B5	B4	B3	B2	B1	B0
シリアル・ステータス・レジスタ 2 ($1A7)	DSR	PBRK	FE	OE	PE	TXE	RX RDY	TX RDY
シリアル・データ・レジスタ 2 ($1A9)	D7	D6	D5	D4	D3	D2	D1	D0

図6.19 シリアル・レジスタ・マップ

6.3.1.1 特徴

完全に独立した3チャネルによる非同期通信をサポート

チャネルごとに割込み要求を発生

5~8ビット・キャラクタ長をプログラム可能

ストップ・ビットは1ビット,2ビット選択可能

パリティ・ビットの付加が可能(パリティ無し、偶数パリティ、奇数パリティ)

パリティ,オーバーランおよびフレーミング・エラーの検出可能

誤りスタート・ビット検出

自動ブレーク検出

ブレーク・キャラクタ送出

全二重通信,ダブル・バッファ方式

転送レート:1Mbps(max)

ボーレート・ジェネレータ内蔵

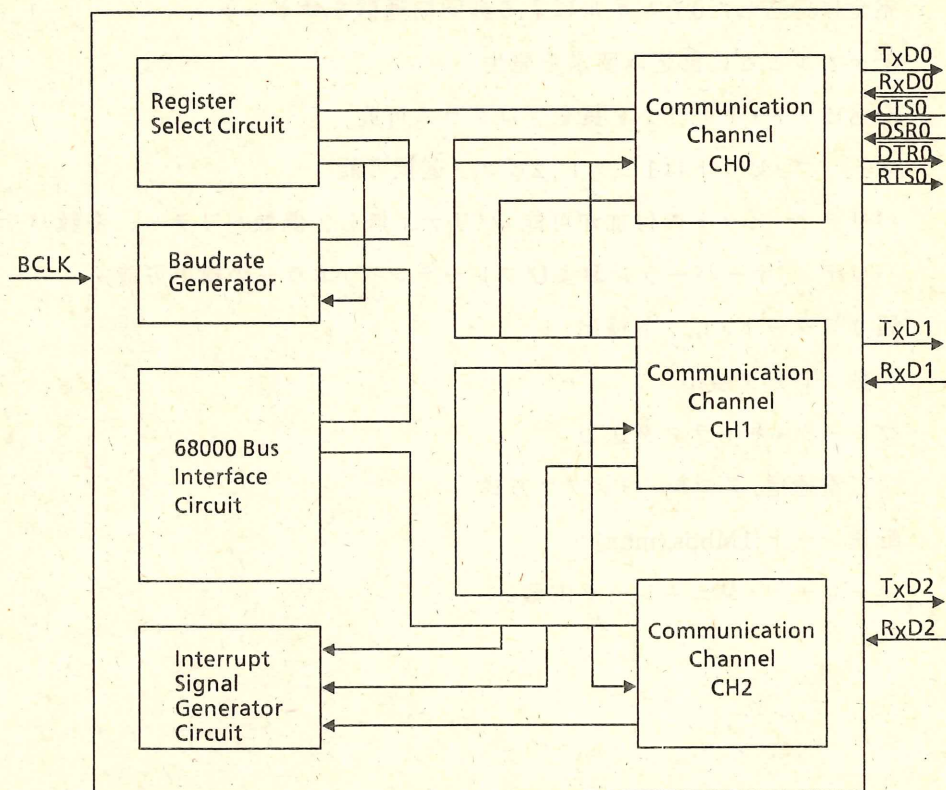



図6.20 シリアル・インタフェース・ブロック図

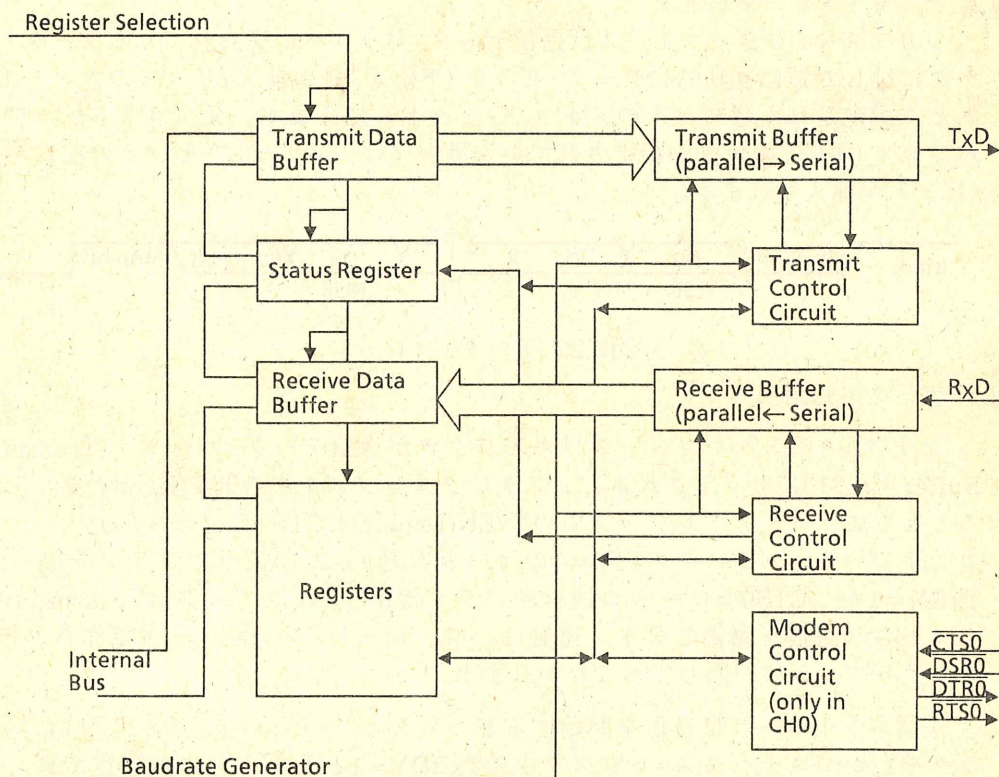



図6.21 チャネル詳細図

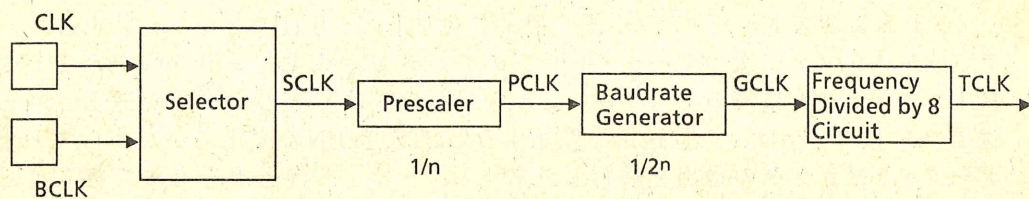



図6.22 ボーレート・ジェネレータ詳細図

6.3.2 通信動作の概要

6.3.2.1 データ・フォーマット

シリアル・インターフェースは送信データ・バッファに送られてくるデータ・キャラクタに対して常に自動的にデータ・ビット(下位が先)の前に1ビットのスタート・ビットと、指定された数ビットのストップ・ビットを加えます。又、ソフトウェアによりパリティ(偶/奇)の付加が指定されている場合には、ストップ・ビットの前にパリティ・ビットが挿入されます。

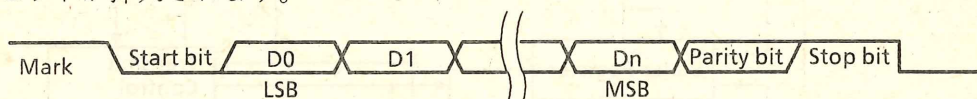



図6.23 データ・フレーム

6.3.2.2 データの送信

シリアル・インターフェースはキャラクタが送信データ・バッファ(Transmit Data Buffer,図6.21)にセットされると、ステータスレジスタ中のTxE(Empty)をリセットするとともに、コマンドレジスタ中のTxEN(Enable)とCTS(チャンネル0で、パラレル・コントロール・レジスタのPF0=1の時のみ有効)および送信中フラグを調べます。TxEN=1かつ$\overline{\text{CTS0}}$=ローかつ送信中フラグが0であればデータはTransmit Bufferにセットされ送信を開始します。送信は、ボーレート・ジェネレータで作られるTCLKの立上りエッジに同期しておこなわれます。

送信コントローラは送信を開始すると、スタート・ビットの送出と同時に送信中フラグを1,かつステータス・レジスタ中のTxRDY=1とします。この時点であらかじめ次に送信するデータ・キャラクタを送信データ・バッファにセットしておくことが可能です。次に送信するデータ・キャラクタがセットされると、現在送信中のデータ・キャラクタを送信し終えるまで送信データ・バッファに保持されます。送信コントローラは、スタート・ビットの送出後、指定されたキャラクタ長のデータにパリティ・ビットおよびストップ・ビットを自動的に付加して、TxDラインに送出します。送信データ・バッファに次のデータ・キャラクタがある時は現在送出されているデータのストップ・ビットに続いてスタート・ビットから続けて次のデータが送出されます。送信コントローラは、一旦送信を開始するとCTS, TxENの変化にかかわらず、現在のデータ・キャラクタの送出を続行し、ストップ・ビットの送出をもって終了とし、次のデータがなければ送信中フラグを0にリセットします。

6.3.2.3 データの受信

RxDラインは通常ハイ・レベル(マーク状態)になっています。この入力に立下がりエッジがはいると、スタート・ビットの始まりを検出します。受信ではまず、ボーレート・ジェネレータで作られるGCLKの立上りでサンプリングした最初のLowレベル検出後、連続した4回のGCLKの立上りによるサンプリング・データがLowレベルである時、最初のLowレベルを有効なスタート・ビットであると判断し、それ以後の

データ・ビットの中心点を決定し、各データ・ビットのサンプリングをTCLKの立下がりエッジで行います。

パリティがある場合、パリティ・ビットの中心をサンプリングし、受信されたデータによって生成されるパリティと比較し、一致がとれなければパリティ・エラー・フラグを1にセットします。また、ストップ・ビットについては、プログラムで指定されているストップ・ビットの数には無関係に、第1のストップ・ビットの中心をサンプリングし、もしそれが1でないならばフレーミング・エラー・フラグを1にセットします。データ・ビットがプログラムされたビット数分までサンプリングされた時点で、それらをまとめて受信データ・バッファに転送し、ステータス・レジスタ中のRxRDYフラグを1にセットします。この時未使用の上位ビットがある場合、それは、0にリセットして転送されます。

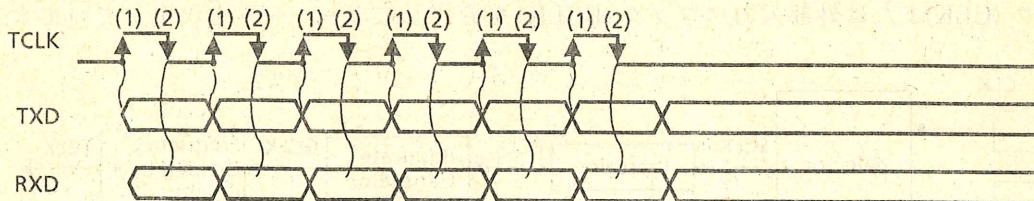
RxRDYフラグは、受信データ・バッファに読み出すべきデータがあることを示します。もし次のデータ・キャラクタのサンプリングが完了して、データが受信データ・バッファに転送されるまでに、前に書込まれたデータを読み出していないと、受信データ・バッファの内容は新しいデータに置換わり、前のデータは消滅し、オーバーラン・エラー・フラグが1にセットされます。また、すべてのデータ・ビット及び、もしあればパリティ・ビットまたはストップ・ビットがすべて0である様なデータが2個以上連続して入力されると受信コントローラは、ブレーク検出フラグを1にセットします。この時、RxDイニシャライズ回路の働きにより、RxDラインが1になるまで次のスタート・ビットの検出は保留されます。ブレーク検出フラグをはじめ全てのエラー・フラグはエラー・リセット(ERS)コマンドによってリセットするまで1を保持します。ただし、エラー発生そのものは受信動作には影響を与えません。

(1)：送信データ・ビットの切換えタイミング
(2)：受信データ・ビットのサンプリング・タイミング

図6.24 データの送受信タイミング図

6.3.3 割込み制御

シリアル・インターフェースにおける割込み要因として次の3つがあります。

1. 送信チャネルにおいて、送信データ・バッファが新しいデータを受け取れる状態になった時
   (送信スタート時は、バッファが空の状態でも割込みは発生しません。一度バッファにデータがセットされた後、次のデータがセットできるようになった時割込みが入ります。スタート時データをセット後、TxENをセットして下

さい。)

2. 受信チャネルにおいて、受信されたデータが受信データ・バッファにあるか、またはブレーク・キャラクタが検出された時

3. 受信チャネルにおいて、エラーが発生した時

これらおのおのは、モード・レジスタにおいて個別にマスクすることができます。マスクされた状態で入った割込み要求は保持され、マスクをとると割込み発生します。また、これらの割込み要因は、受信側の要因はRxEN, 送信側の要因はTxENがそれぞれ条件となっています。

さらに要因の個別なマスクとは別に、シリアル・インターフェースとして割込み信号の発生を禁止/許可することがコントロール・レジスタにおいて可能です。

割込み＝(INTM＝0)×{(送信割込み)＋(受信割込み)}

送信割込み＝(TxEN＝1)×($\overline{\text{CTS0}}$＝ロー)×(TxRDY＝1)×(TxINTM＝0)

(注)$\overline{\text{CTS0}}$を使用しない時は$\overline{\text{CTS0}}$は無視されます。$\overline{\text{CTS0}}$の使用/不使用はパラレル・コントロール・レジスタ(6.4.3.2.)により決まります。

受信割込み＝(RxEN＝1)[(RxINTM＝0)×{(RxRDY＝1)＋(RBRK＝1)}

＋(ERINTM＝0)×(FE＋OE＋PE)]

6.3.4 ボーレートの生成

シリアル・インターフェースはボーレート・ジェネレータ・ブロックにおいて、8ビットのプリスケーラと8ビットのボーレート・ジェネレータとでシステム・クロック(CLK)または外部入力クロック(BCLK)を分周してボーレートを決定しています。

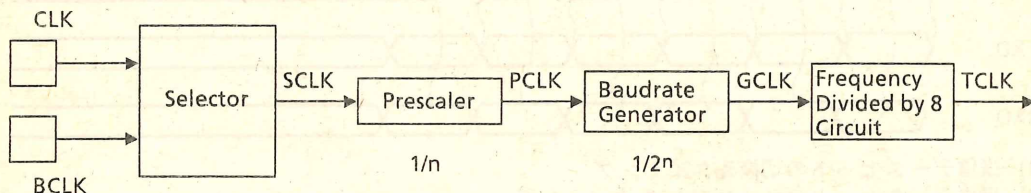



図6.25 ボーレート・ジェネレータ・ブロック

まず、セレクタでシステム・クロックとBCLKのどちらを使うのか選択し、その選択したSCLKをプリスケーラにおいて×1～×1/256に分周し、この分周したPCLKをボーレート・ジェネレータにおいて更に×1～×1/128に1/(2のn乗)毎に分周し、この分周したGCLKを送受信コントローラの制御用クロックとして使用し、このGCLKを更に8分周したTCLKをデータ・ビットのシフト・アウトならびにサンプリング用クロックとして使用します。

6.3.5 システム・リセットとイニシャライズ

シリアル・インターフェースのイニシャライズは、システム・リセットを掛けるかまたは、プログラムによってコントロール・レジスタのビット5 (RES) を1にする事で可能です。上述の処理によって、シリアル・インターフェースのシステム・リセット・フリップ・フロップがセットされ、次のコントロール・レジスタへの書込みにおいて、ビット5を0にプログラムするまでこの状態を維持します。これは、電源投入後、初期設定プログラムが必要とするイニシャライズ値をセットするまで不要な動作を規制する事に利用できます。システム・リセットによってイニシャライズされるのは、次のレジスタと内部ロジックの一部です。レジスタのイニシャライズは次の通りです。

Register	7	6	5	4	3	2	1	0
Control	CKSE	–	RES	–	–	–	–	INTM
Register	1	×	1	×	×	×	×	1
Command	–	–	RTS	ERS	SBRK	R$_X$EN	DTR	T$_X$EN
Register	×	×	0	1	0	0	0	0
Status	DSR	RBRK	FE	OE	PE	T$_X$E	R$_X$RDY	T$_X$RDY
Register	external	0	0	0	0	1	0	×

6.3.6 内部レジスタ

6.3.6.1 シリアルコントロール・レジスタ

シリアル・インターフェースの基本的な設定をするレジスタです。

bit 0：割込み信号発生の禁止/許可を指定 (INTM)
　1：割込み信号発生禁止
　0：割込み信号発生許可

bit 5：ソフトウェア・リセット (RES)
　1：リセット
　0：解除

bit 7：分周するクロックの選択 (CKSE)
　1：システム・クロック
　0：BCLK

各レジスタの設定は、bit5のソフトウェア・リセットを解除してから行って下さい。

7	6	5	4	3	2	1	0
CKSE	×	RES	×	×	×	×	INTM

(オフセット　$18F)

図6.26 シリアル・コントロール・レジスタ

(注)　将来の拡張性のため、“×”ビットには0をセットして下さい。

6.3.6.2 シリアルモード・レジスタ0~2

チャネル毎に持っており、キャラクタの構成や割込み発生のマスクに関するレジスタです。

bit 0: ストップ・ビット数を指定 (ST)
1: 2ストップ・ビット
0: 1ストップ・ビット

bit 1: TxRDYでの割込みを制御 (TxINTM)
1: 割込みマスク
0: 割込み有効

bit3, 2: キャラクタ長を指定 (CL1, CL0)
0, 0: 5ビット・キャラクタ
0, 1: 6ビット・キャラクタ
1, 0: 7ビット・キャラクタ
1, 1: 8ビット・キャラクタ

bit 4: パリティ制御 (PEN)
1: パリティ有り
0: パリティ無し

bit 5: パリティ選択 (PEO)
1: 奇数パリティ
0: 偶数パリティ

bit 6: 受信エラーによる割込みを制御 (ERINTM)
1: 割込みマスク
0: 割込み有効

bit 7: キャラクタ受信による割込みを制御 (RxINTM)
1: 割込みマスク
0: 割込み有効

7	6	5	4	3	2	1	0
Rx INTM	ER INTM	PEO	PEN	CL1	CL0	Tx INTM	ST

(オフセット $181 ch0, $191 ch1, $1A1 ch2)

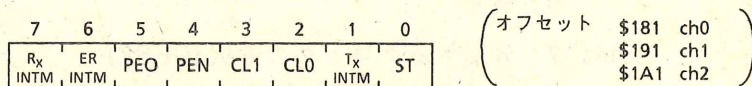



図6.27 シリアル・モード・レジスタ

6.3.6.3 シリアル・コマンド・レジスタ0~2

チャネル毎に持っており、送/受信の制御線のコントロールに関するレジスタです。

bit 0：送信状能 (TxEN)
　1：送信可能
　0：送信停止

bit 1：出力端子DTRの制御(チャネル0のみ) (DTR)
　1：$\overline{\text{DTR0}}$ピンはロー
　0：$\overline{\text{DTR0}}$ピンはハイ

bit 2：受信状態を指定 (RxEN)
　1：受信可能
　0：受信停止

bit 3：ブレーク・キャラクタの送出を指定(SBRK)
　1：送出
　0：停止

bit 4：受信エラー・フラグをリセット (ESR)
　1：PE、OE、FE、RBRKフラグをリセット
　0：ノー・オペレーション

bit 5：出力端子RTSの制御 (チャネル0のみ) (RTS)
　1：$\overline{\text{RTS0}}$ピンはロー
　0：$\overline{\text{RTS0}}$ピンはハイ

7	6	5	4	3	2	1	0
×	×	RTS	ESR	SBRK	R_XEN	DTR	T_XEN

(オフセット　$183 ch0
　　　　　　$193 ch1
　　　　　　$1A3 ch2)

図6.28 シリアル・コマンド・レジスタ

(注)将来の拡張性のため、“×”のビットには0をセットして下さい。

6.3.6.4 シリアル・ステータス・レジスタ0~2

チャネル毎に持っており、チャンネルの状態を示すレジスタです。

bit 0: 送信レディの状態を示す(TxRDY)
{(送信データ・バッファが空)×($\overline{\text{CTS0}}$=ロー)×(TxEN=1)}で1

(注) チャネル1,2及び$\overline{\text{CTS0}}$を使用しない場合は、$\overline{\text{CTS0}}$=ローと同じです。$\overline{\text{CTS0}}$使用/不使用は6.4.3.2パラレル・コントロール・レジスタで設定して下さい。
送信割込みを使用している場合は割込みアクノリジサイクルで0になります。

bit 1: 受信バッファの状態を示す(RxRDY)
受信バッファに受信されたキャラクタが有る時に"1"

bit 2: 送信バッファの状態を示す(TxE)
送信バッファが空で、かつ、送信中でない時に"1"

bit 3: パリティ・エラー・ビット(PE)
受信データのパリティ・エラーが検出されると"1"

bit 4: オーバー・ラン・エラー(OE)
受信バッファ中の受信データが読み出される事なく、次のキャラクタ受信によってデータが消滅すると"1"

bit 5: フレーミング・エラー(FE)
1つのキャラクタ受信において、ストップ・ビットが検出されないと"1"

bit 6: ブレーク・キャラクタ検出(PBRK)
受信中にブレーク・キャラクタを検出すると"1"

bit 7: 入力端子DSRのセンス(チャンネル0のみ)(DSR)
$\overline{\text{DSR0}}$ピンの入力レベルがローの時"1"

7	6	5	4	3	2	1	0
DSR	PBRK	FE	OE	PE	T_XE	R_X RDY	T_X RDY

(オフセット $187 ch0, $197 ch1, $1A7 ch2)

図6.29 シリアル・ステータス・レジスタ

6.3.6.5 シリアル・プリスケーラ・レジスタ

プリスケーラでの分周比を決定するレジスタです。

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

(オフセット $18D)

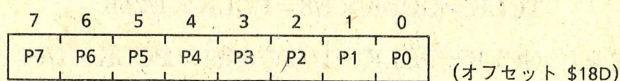



図6.30 シリアル・プリスケーラ・レジスタ

このレジスタにセットされる値は0~255までの値です。おのおのの値による分周比は次の通りです。

セット値	分周比
0	×1/256
1	×1
2~255	×1/2~×1/255

6.3.6.6 ボーレイト・レジスタ0~2

チャネル毎に持っており、プリスケーラで分周されたクロック(PCLK)を更にどのくらい分周したクロックを使用するのかを選択するレジスタで、一度に複数のビットをセットすることはできません。ただし、このレジスタで選択したPCLKを分周したクロック(GCLK)は送受信コントローラを制御するのに使われ、実際にデータ・ビットのシフト・アウトおよびサンプリングを行うために使われるクロック(TCLK)はGCLKを更に8分周した値になります。

7	6	5	4	3	2	1	0
B7	B6	B5	B4	B3	B2	B1	B0

(オフセット $185 ch0 / $195 ch1 / $1A5 ch2)

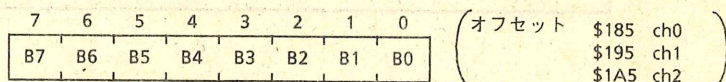



図6.31 ボーレイト・レジスタ

B0=1　GCLK=PCLK×1/(2の0乗)=PCLK×1

TCLK=GCLK×1/8=PCLK×1/8

B1=1　GCLK=PCLK×1/(2の1乗)=PCLK×1/2

TCLK=GCLK×1/8=PCLK×1/16

B2=1　GCLK=PCLK×1/(2の2乗)=PCLK×1/4

TCLK=GCLK×1/8=PCLK×1/32

B3=1　GCLK=PCLK×1/(2の3乗)=PCLK×1/8

TCLK=GCLK×1/8=PCLK×1/64

B4=1　GCLK=PCLK×1/(2の4乗)=PCLK×1/16

TCLK＝GCLK×1/8＝PCLK×1/128

B5＝1　GCLK＝PCLK×1/(2の5乗)＝PCLK×1/32

TCLK＝GCLK×1/8＝PCLK×1/256

B6＝1　GCLK＝PCLK×1/(2の6乗)＝PCLK×1/64

TCLK＝GCLK×1/8＝PCLK×1/512

B7＝1　GCLK＝PCLK×1/(2の7乗)＝PCLK×1/128

TCLK＝GCLK×1/8＝PCLK×1/1024

プリスケーラレジスタとボーレートレジスタで指定されるボーレートの一例(BCLKの周波数 f＝1.8432MHz)を示します。

f = 1.8432MHz

ボーレートレジスタ ＼ プリスケーラの値	6	20	48
B0 = 1	38.4K	11.52K	4800
B1 = 1	19.2K	5760	2400
B2 = 1	9600	2880	1200
B3 = 1	4800	1440	600
B4 = 1	2400	720	300
B5 = 1	1200	360	150
B6 = 1	600	180	75
B7 = 1	300	90	

6.3.6.7 シリアル・データ・レジスタ0－2

チャネル毎に持っており、送信するデータや受信したデータを一時保持するレジスタです。

送信データ・レジスタと受信データ・レジスタのレジスタ・アドレスは同じですが、リード/ライト信号と組合せることによって区別しています。

	7	6	5	4	3	2	1	0
シリアル・データ・レジスタ	D7	D6	D5	D4	D3	D2	D1	D0

(オフセット　$189 ch0
$199 ch1
$1A9 ch2)

図6.32 シリアル・データ・レジスタ

6.4 パラレル・インターフェース

6.4.1 概要

このパラレル・インターフェースは、汎用の16ビットのI/Oポートを持ち、1ビット単位でのI/O指定が可能で、モードの切り替えによってセントロニクス・インターフェースに対応できます。

セントロニクス・インターフェースでは、16ビット・データ・バスが8ビット・データ・バスと最大5本の制御信号になります。セントロニクス送信時では8ビット・データを受け取ると、予め内部レジスタにプログラムされた時間で外部へデータ・ストローブを発生します。セントロニクス受信時では外部からのデータ・ストローブで8ビットのデータを内部に保持し、BUSY信号の応答を返します。またセントロニクス送信/受信時において合計3つの割込みを発生します。

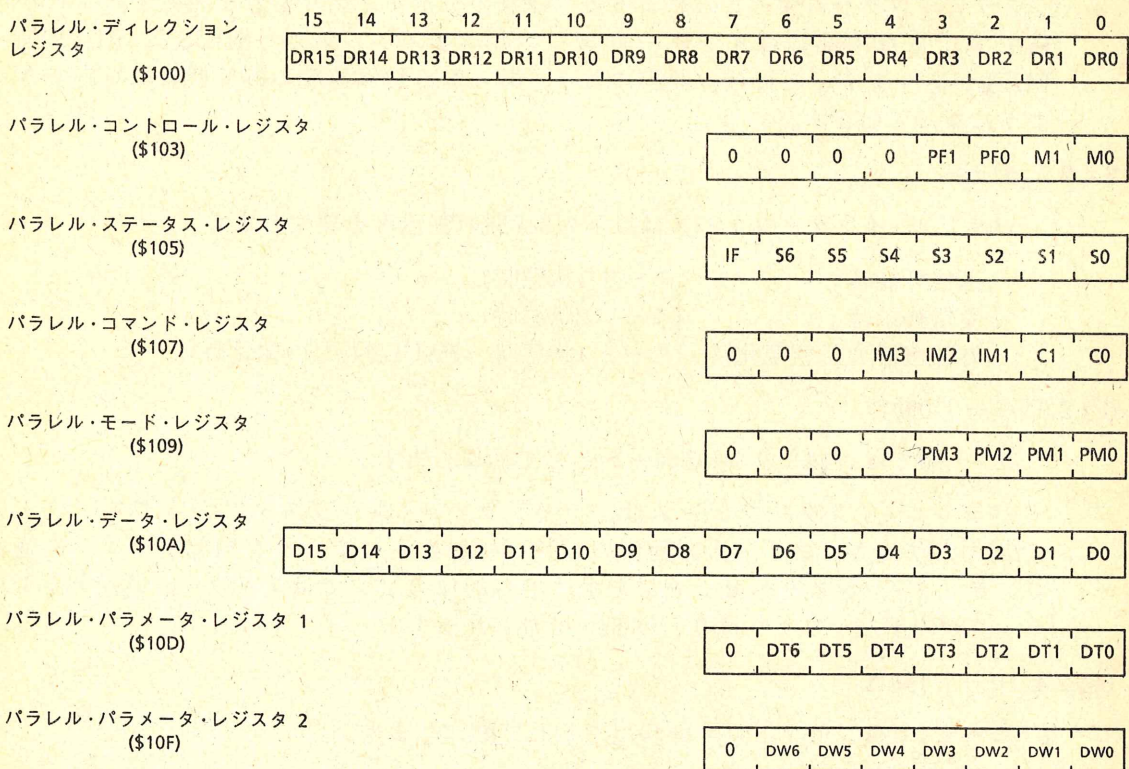



図6.33 パラレル・レジスタ・マップ

6.4.2 動作モード

パラレル・インターフェースは、動作モードとして3つのモードを持ち、その内の1つを内部のコントロール・レジスタによって選択します。各モードでは、数種のピン・ファンクションが選択できるようになっており、パラレル・インターフェースのI/Oポートの一部をシリアル・インターフェースとして、またセントロニクス対応の制御信号として使用することができます。

モード0	16ビット・バス(入/出力動作)
モード1	8ビット・バス(出力動作)+制御信号(セントロニクス出力)
モード2	8ビット・バス(入力動作)+制御信号(セントロニクス入力)

6.4.2.1 制御信号の自動発生機能

パラレル・インターフェースは、セントロニクス制御信号のうち$\overline{\text{DSTB}}$、$\overline{\text{ACK}}$をレジスタにプログラムすることによって、任意のタイミングで信号を発生することが可能です。また内部にはインターフェース用に2つのフラグ、XBUSY、BUFFER-FULLがあります。これらの機能により、あらゆる動作速度の外部機器に対応できるようになっています。

6.4.2.2 割込み

パラレル・インターフェースは以下の3状態の割込みを発生します。

送信可能時	(モード1状態時)
受信終了時	(モード2状態時)
外部機器の状態変化時	(モード1状態時…FAULT入力の変化時)

6.4.2.3 モード0動作

モード0では、16ビットI/Oポートとして動作します。

パラレル・インターフェースは、ディレクション・レジスタによって各ビットの入/出力が決定されます。入力動作のピンは外部からのデータを内部バッファを通して、データ・レジスタへセットします。出力動作のピンではデータ・レジスタからのデータが内部バッファを通り、外部へ出力されます。

6.4.2.4 モード1動作

モード1では、セントロニクス出力として動作します。

パラレル・インターフェースは、コア・プロセッサからのデータを受け取ると、内部バッファに保持すると共に、パラメータ・レジスタ1、2で設定されたタイミングで$\overline{\text{DSTB}}$を発生します。この後、外部が返すBUSY信号のネゲートを受けて割込みを発生します。もし、外部のBUSY信号の遅れを補いたいときの為に、内部にXBUSYフラグを用意しています。このXBUSYフラグは内部バッファにデータが保持されると同時にセットされ、外部からの$\overline{\text{ACK}}$信号を受けてリセットされるものです。また、モード・レジスタによって割込みの発生タイミングを外部BUSY信号の立ち下がり

か、内部XBUSYフラグがクリアされたときのいずれかを選ぶことができます。

ピン・ファンクションでPRIMEとFAULTを追加した場合には、PRIMEが出力ピンにFAULTが入力ピンになり、それぞれプリンターの初期化信号と異常検出信号に用いられます。

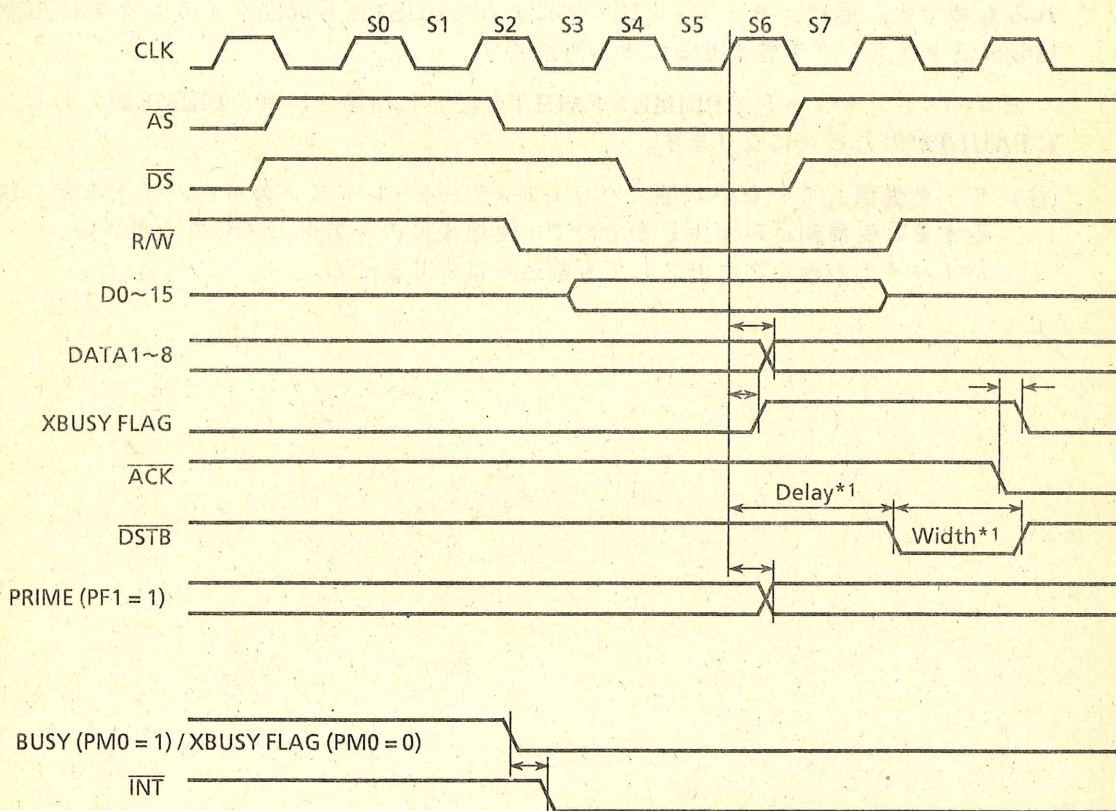



図6.34 セントロニクス出力タイミング

6.4.2.5 モード2動作

モード2では、セントロニクス入力として動作します。

パラレル・インターフェースは、外部からのデータ・ストローブを受けて、内部バッファにデータを保持すると同時に、割込みと外部へのBUSY信号を発生します。この後、コア・プロセッサからのリード,ダミー・ライトによって、パラメータ・レジスタ1、2で設定されたタイミングで外部へ$\overline{\text{ACK}}$信号を返します。また、内部にはデータを受信したことを示すBUFFER−FULLフラグを用意しています。これは、内部バッファにデータが保持されるとセットされ、外部への$\overline{\text{ACK}}$信号の発生でリセットされるものです。更に、モード・レジスタによってBUSY信号成立タイミング中にACK信号成立タイミングを含ませることも可能です。

ピン・ファンクションでPRIMEとFAULTを追加した場合には、PRIMEが入力ピンにFAULTが出力ピンになります。

(注) データ受信していない状態でパラレル・データ・レジスタのローバイトをから読みすると受信割込みが生じますので、受信後にのみアクセスしてください。
ハイバイトのみをアクセスしても割込みは生じません。

MODE2 (CENTRONICS INPUT MODE)

S0 S1 S2 S3 S4 S5 S6 S7

CLK
AS
DS
R/W
D0~15
DSTB (PM3 = 0)
DSTB (PM3 = 1)
DATA1~8
ACK (PM2 = 1)
BUSY (PM1 = 0)
BUSY (PM1 = 1)
BUFFER FULL FLAG
INT

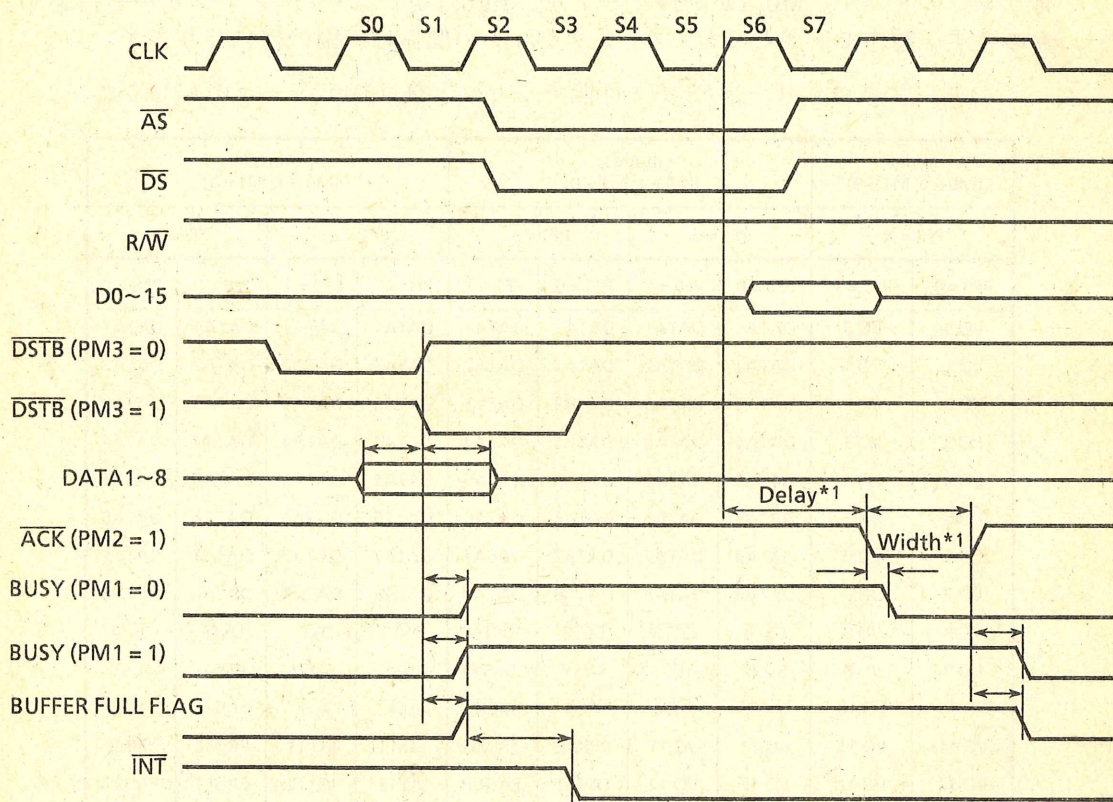
Delay*1
Width*1

*1 : 1CLK~128CLK

図6.35 セントロニクス入力タイミング

6.4.3 レジスタ構成

6.4.3.1 パラレル・デイレクション・レジスタ

パラレル・デイレクション・レジスタは、16ビット・ポートの各ビットに対する入/出力をプログラムするレジスタで、"1"で出力、"0"で入力として動作します。

このレジスタは、リード/ライト可能でリセット後は$0000(16ビット入力状態)になります。

図6.36 パラレル・ディレクション・レジスタ

6.4.3.2 パラレル・コントロール・レジスタ

パラレル・コントロール・レジスタは、動作モードとピン・ファンクションの選択を行うレジスタです。M0, 1が動作モードを、PF0, 1がピン・ファンクションを指定するものです。動作モードとピン・ファンクションの関係は下図の様になります。

このレジスタは、リード/ライト可能でリセット後は$00(モード0)になります。

mode0 (M0 = 0, M1 = 0)		mode1 (M0 = 1, M1 = 0)				mode2 (M0 = X, M1 = 1)			
PF1 = X		PF1 = 0		PF1 = 1		PF1 = 0		PF1 = 1	
PF0 = 0	PF0 = 1	PF0 = 0	PF0 = 1	PF0 = 0	PF0 = 1	PF0 = 0	PF0 = 1	PF0 = 0	PF0 = 1
I/O 0	I/O 0	DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA1
I/O 1	I/O 1	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2
I/O 2	I/O 2	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3
I/O 3	I/O 3	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4
I/O 4	I/O 4	DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA5
I/O 5	I/O 5	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6
I/O 6	I/O 6	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7
I/O 7	I/O 7	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8
I/O 8	I/O 8	$\overline{DSTB}$	$\overline{DSTB}$	$\overline{DSTB}$	$\overline{DSTB}$	$\overline{DSTB}$	$\overline{DSTB}$	$\overline{DSTB}$	$\overline{DSTB}$
I/O 9	I/O 9	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY
I/O 10	I/O 10	$\overline{ACK}$	$\overline{ACK}$	$\overline{ACK}$	$\overline{ACK}$	$\overline{ACK}$	$\overline{ACK}$	$\overline{ACK}$	$\overline{ACK}$
I/O 11	I/O 11	I/O 11	I/O 11	PRIME	PRIME	I/O 11	I/O 11	PRIME	PRIME
I/O 12	I/O 12	I/O 12	I/O 12	FAULT	FAULT	I/O 12	I/O 12	FAULT	FAULT
I/O 13	$\overline{CTS0}$	I/O 13	$\overline{CTS0}$	I/O 13	$\overline{CTS0}$	I/O 13	$\overline{CTS0}$	I/O 13	$\overline{CTS0}$
I/O 14	$\overline{DSR0}$	I/O 14	$\overline{DSR0}$	I/O 14	$\overline{DSR0}$	I/O 14	$\overline{DSR0}$	I/O 14	$\overline{DSR0}$
I/O 15	$\overline{DTR0}$	I/O 15	$\overline{DTR0}$	I/O 15	$\overline{DTR0}$	I/O 15	$\overline{DTR0}$	I/O 15	$\overline{DTR0}$

図6.37 パラレル・コントロール・レジスタ

6.4.3.3 パラレル・ステータス・レジスタ

パラレル・ステータス・レジスタは、モード1,2でのセントロニクス動作における各制御信号の内容が保持されます。但し、S0,1,4,5が有効になるのはコントロール・レジスタのPF1が“1”のときだけです。S0~3はモード1の制御信号の内容が保持され、S4~6はモード2の制御信号の内容が保持されます。IFはモード1,2状態で割込みが発生すると“1”がセットされます。セットされたフラグは割込みアクノリッジ・サイクル実行後“0”をセットするか、リセットが入力されるまで保持されます。

このレジスタは、リードでき、IFのみライト可能です。

S0 : FAULT入力
S1 : PRIME出力
S2 : BUSY入力
S3 : XBUSYフラグ
S4 : FAULT出力
S5 : PRIME入力
S6 : BUFFER−FULLフラグ
IF : 割込みフラグ

図6.38 パラレル・ステータス・レジスタ

6.4.3.4 パラレル・コマンド・レジスタ

パラレル・コマンド・レジスタは、モード1,2での出力ピンの状態設定と3つの割込みの有効/無効を選択するレジスタです。割込み無効中に入った割込み要求は無視され、保持されません。割込みを先へのばしたい場合はインタラプトコントローラのマスクレジスタを使用して下さい。

C0,IM1,2はモード1の場合に、C1,IM3はモード2の場合に有効になります。但し、C0,1,IM2が有効になるのは、コントロール・レジスタのPF1が“1”のときだけです。

このレジスタは、リード/ライト可能でリセット後は$00になります。

C0=0 : PRIME出力のレベルを“0”レベルにします。
C0=1 : PRIME出力のレベルを“1”レベルにします。
C1=0 : FAULT出力のレベルを“0”レベルにします。
C1=1 : FAULT出力のレベルを“1”レベルにします。

IM1=1 : 送信可能時の割込みを無効にします。
IM2=1 : 外部機器の状態変化時の割込みを無効にします。
IM3=1 : 受信終了時の割込みを無効にします。

図6.39 パラレル・コマンド・レジスタ

6.4.3.5 パラレル・モード・レジスタ

パラレル・モード・レジスタは、BUSY信号と割込み、BUSY信号と$\overline{ACK}$信号の関係を定義するレジスタです。モード1の場合PM0がモード2の場合PM1, 2, 3が有効になります。

このレジスタは、リード/ライト可能でリセット後は$00になります。

PM0＝0 ： XBUSYフラグがクリアされた時に転送可能割込みを発生します。
PM0＝1 ： 外部BUSYの立ち下がりで転送可能割込みを発生します。
PM1＝0 ： BUSY信号には$\overline{ACK}$が含まれません。
PM1＝1 ： BUSY信号には$\overline{ACK}$が含まれます。
PM2＝0 ： ダミー・ライトでBUSYがリセットされ$\overline{ACK}$が発生します。
(これを使って$\overline{ACK}$アサートをソフトで遅らせることができます。)
PM2＝1 ： リードでBUSYがリセットされ$\overline{ACK}$が発生します。
PM3＝0 ： $\overline{DSTB}$の立ち上がりでデータを取込みます。
PM3＝1 ： $\overline{DSTB}$の立ち下がりでデータを取込みます。

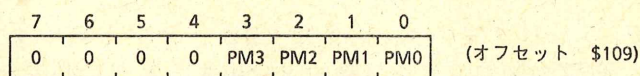



図6.40 パラレル・モード・レジスタ

6.4.3.6 パラレル・データ・レジスタ

パラレル・データ・レジスタは、I/Oポートからの入/出力データの読出し/書込みに使用されるレジスタです。

I/Oポートを入力状態として使用した場合、またはセントロニクス受信として使用した場合は、外部からのデータが受信バッファに取り込まれると、同時にこのレジスタにセットされます。但し、この場合にデータを書込んでもデータは無視されます。

I/Oポートを出力状態として使用した場合、またはセントロニクス送信として使用した場合は、このレジスタの対応するビットに出力するデータを書込むと、外部へ出力されます。この場合にデータを読込むと、ポートに出力しているデータが読みこまれます。

このレジスタは、リセット後$0000(入力状態)になります。

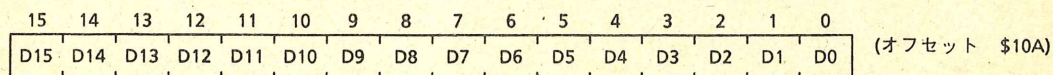



図6.41 パラレル・データ・レジスタ

6.4.3.7 パラレル・パラメータ・レジスタ1

パラレル・パラメータ・レジスタ1は、セントロニクス・インターフェースでの$\overline{\text{DSTB}}$, $\overline{\text{ACK}}$の制御信号を自動的に発生するときに必要な時間をプログラムするレジスタです。セットされた値で$\overline{\text{DSTB}}$, $\overline{\text{ACK}}$のDelayを自動発生します。モード1状態時では$\overline{\text{DSTB}}$のDelayのクロック数、モード2状態時では$\overline{\text{ACK}}$のDelayのクロック数を示します。

このレジスタは、リード/ライト可能でリセット後は$00になります。

Delay = DT6 × 64 + DT5 × 32 + DT4 × 16 + DT3 × 8 + DT2 × 4 + DT1 × 2 + DT0

7	6	5	4	3	2	1	0
0	DT6	DT5	DT4	DT3	DT2	DT1	DT0

(オフセット　$10D)

図6.42 パラレル・パラメータ・レジスタ1

6.4.3.8 パラレル・パラメータ・レジスタ2

パラレル・パラメータ・レジスタ2は、セントロニクス・インターフェースでの$\overline{\text{DSTB}}$、$\overline{\text{ACK}}$の制御信号を自動的に発生するときに必要な時間をプログラムするレジスタです。セットされた値で$\overline{\text{DSTB}}$、$\overline{\text{ACK}}$のWidthを自動発生します。モード1状態時で$\overline{\text{DSTB}}$のWidthのクロック数、モード2状態時では$\overline{\text{ACK}}$のWidthのクロック数を示します。上記のパラメータ・レジスタ1と組み合せて使用することになります。

このレジスタは、リード/ライト可能でリセット後は$00になります。

Width = DW6 × 64 + DW5 × 32 + DW4 × 16 + DW3 × 8 + DW2 × 4 + DW1 × 2 + DW0

7	6	5	4	3	2	1	0
0	DW6	DW5	DW4	DW3	DW2	DW1	DW0

(オフセット　$10F)

図6.43 パラレル・パラメータ・レジスタ2

6.5 タイマ

6.5.1 概要

このタイマは独立した3チャネルの16ビット・カウンタで構成されており、チャネルごとに8ビット・プリスケーラ(システムクロック使用時のみ有効)を持ちます。さらにチャネルごとに割込み要求を発生します。またこのタイマは3チャネルをカスケード接続でき、長い周期のカウントが可能です。

独立した3チャネルの16ビット・カウンタ内蔵
カウントデータの読出し可能
チャネルごとに割込みを発生
ソフトウェア/ハードウェア・トリガ
プログラム可能な動作モード
任意のデューティ比波形出力(チャネル1,2のみ)
ワンショット・パルス出力(チャネル1,2のみ)
イベント・カウント
最大8MHzカウント

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
タイマ・コントロール レジスタ 0 [$200]	CK2	CK1	P4	P3	P2	P1	T2	T1	N/1	1	0	1	0	INT	CS	TS
MAX カウント・レジスタ 01 [$204]	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
タイマ・カウント・レジスタ 0 [$20C]	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
タイマ・コントロール レジスタ 1 [$220]	CK2	CK1	P4	P3	P2	P1	T2	T1	N/1	R/P	MR2	MR1	0	INT	CS	TS
MAX カウント・レジスタ 11 [$224]	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
MAX カウント・レジスタ 12 [$228]	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
タイマ・カウント・レジスタ 1 [$22C]	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
タイマ・コントロール レジスタ 2 [$240]	CK2	CK1	P4	P3	P2	P1	T2	T1	N/1	R/P	MR2	MR1	0	INT	CS	TS

図6.44 タイマ・レジスタ・マップ

図6.45 タイマ・ブロック図

6.5.2 動作説明

6.5.2.1 チャネル0~2

各チャネルの内部カウンタは16ビット・アップ・カウンタです。16ビット・カウンタのカウント値はMAXカウント・レジスタ値に達すると同時にカウント一致信号を発生しカウンタ値をクリアします。チャネル内で、割込み要求信号を発生できます。

6.5.2.1.1 チャネル0

チャネル0は3個のレジスタを持っています(コントロール・レジスタ(16ビット),カウント・レジスタ(16ビット),MAXカウント・レジスタ(16ビット)。このチャネルは外部入力端子(チャネル0~2共通)より外部クロック又は外部信号を取込むことが可能で、外部イベントのカウントができます。

6.5.2.1.2 チャネル1,2

チャネル1,2は4個のレジスタを持っています(MAXカウント・レジスタ1及び2、コントロール・レジスタ,カウント・レジスタ)。これらのチャネルは外部入力端子(チャネル0~2共通)および外部出力端子を持っており、外部イベントのカウントおよび任意の波形の出力が可能です。

6.5.2.2 最大カウント数の設定,および変更

MAXカウント・レジスタに書込まれた値が最大カウント数を示します。

MAXカウント・レジスタの値を書換えることで、最大カウント数は設定,および変更されます。(動作中での書換えも可能)

6.5.2.3 カウント値の読出し

カウント中、カウント・レジスタよりカウント値を読出すことが可能です。

カウント・レジスタは読出し専用です。

6.5.3 8ビット・プリスケーラ

システムクロックを、2, 4, 8, 16, 32, 64, 128, 256に分周して、カウントクロックとして使用可能です。分周比はコントロール・レジスタにより設定されます。

6.5.4 割込みの発生

6.5.4.1 割込み発生

割込み発生モードはコントロール・レジスタで制御します。
割込み要求信号を発生しないモードになっている間に発生したカウント一致信号は無視され、保持されませんので、モードを切り換えても割込みは発生しません。

6.5.4.2 割込み発生のタイミング

カウント一致信号の発生時に割込み要求信号が発生します。

6.5.5 レジスタの読出し/書込み

各レジスタの読出し動作は、カウント動作中でもカウント動作や制御信号に影響をあたえません。

カウント動作中にレジスタの書込み動作を行う場合、割込み動作と同一チャネルのカウント一致信号(カウント値とMAXカウント・レジスタ値の比較一致信号)は書込み動作の間無視されます。

6.5.6 タイマの動作モード

6.5.6.1 TIN端子の機能

外部入力端子(TIN)は、外部クロック、トリガ信号、制御信号の入力が可能です。

TINより入力される信号は、全てシステム・クロックの立上がりでサンプリングされています。TINより入力されるクロックおよび、パルスは、システム・クロックの1周期以上のパルス幅のものを使用します。

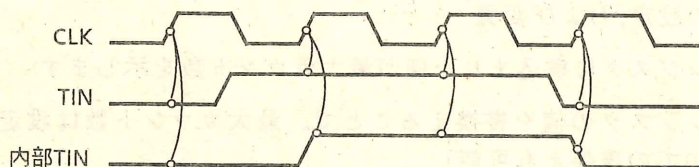



図6.46 TINタイミング

6.5.6.1.1 外部クロック

外部信号が直接カウント・クロックになります。

6.5.6.1.2 カウント制御信号

カウント・コントロール機能として、カウント・スタート機能、カウント・ウエイト機能があり、コントロール・レジスタにより選択されます。

カウント・スタート機能

外部入力端子(TIN)よりトリガ信号を入力します。トリガ信号がLowレベルになった時(立下がり)にカウントがスタートします。(2度め以降の入力は無視されます。)

カウント・ウエイト機能

外部入力端子(TIN)より制御信号を入力します。制御信号がLowレベル状態の間カウント動作が行われます。制御信号がHighレベルの時はカウント動作が停止し、Lowレベルになった時カウントが再開されます。(カウンタがクリアされなければ、停止前のカウント値よりカウントが始まります。)

6.5.6.2 カウント・クロックの選択

最大カウント周波数　8MHz

カウント・クロックは、システム・クロック、外部クロックおよび他のカウンタ出力の内いずれかを選択します。カウント・クロックの選択はカウンタが停止しているときにのみ可能です。

6.5.6.3 TOUT端子の機能(チャネル1, 2)

外部出力端子(TOUT)は、パルス、または矩形波が出力されます。カウント一致時にパルスを発生するか、出力レベルを反転するかを選ぶことができ、2個のMAXカウントレジスタを使えば、任意の矩形波を発生できます。

TOUTから出力される信号(パルスの発生、出力レベル反転のタイミング)はシステム・クロックの立上がりに同期しています。パルス発生の場合は、通常ローでカウント一致時1クロック幅ハイとなります。リセット後はTOUTはローになっています。

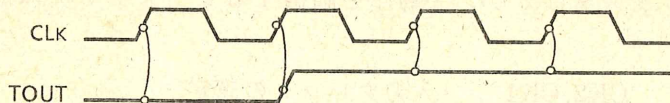



図6.47 TOUTタイミング

6.5.6.4 MAXカウント・レジスタの設定

各チャネルには2個のMAXカウント・レジスタ(以下MAX1, MAX2とします)があり、MAX1のみ、MAX2のみの使用またはMAX1, MAX2を交互に使用できます。MAXカウント・レジスタの使用方法はコントロール・レジスタで設定されます。

6.5.7 レジスタ構成

6.5.7.1 タイマ・カウント・レジスタ

このレジスタは16ビットで構成されています。

カウント値はこのレジスタから読出されます。上位バイトのみ、下位バイトのみの読出しも可能です。このレジスタは読出し専用で書込みを行ってもカウンタには影響をあたえません。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

(オフセット　$20C ch0 / $22C ch1 / $24C ch2)

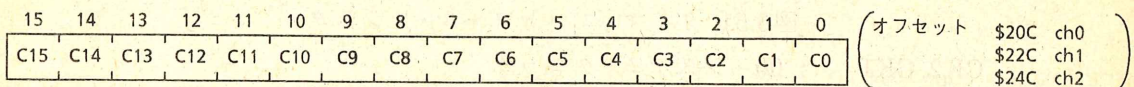



図6.48 タイマ・カウント・レジスタ

6.5.7.2 MAXカウント・レジスタ

このレジスタは16ビットで構成されています。

最大カウント数はMAXカウント・レジスタに書込まれた値になります。

このレジスタは読出し/書込みが可能です。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0

(オフセット $204 ch0 / $224 ch1 / $244 ch2)

図6.49 タイマ・MAXカウント・レジスタ1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0

(オフセット $228 ch1 / $248 ch2)

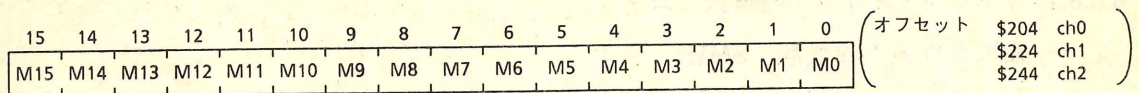
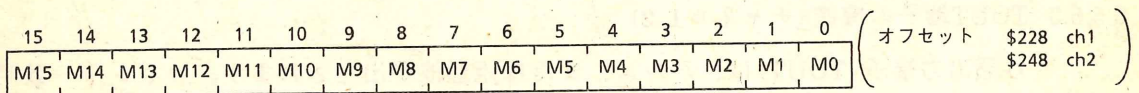



図6.50 タイマ・MAXカウント・レジスタ2

6.5.7.3 タイマコントロール・レジスタ

このレジスタは16ビットで構成されています。

コントロール・レジスタはカウント動作を制御するレジスタで次の様なビットを持っています。

ビット15～14	CK2, CK1	:入力クロックの設定
ビット13～10	P4, P3, P2, P1	:プリスケーラ値の設定
ビット 9～8	T2, T1	:外部信号の選択
ビット 7	N/1	:くりかえし指定
ビット 6	R/P	:出力信号の制御(チャネル1, 2のみ)
ビット 5～4	MR2, MR1	:MAXカウント・レジスタの設定 (チャネル1, 2のみ)
ビット 2	INT	:割込み要求ビット
ビット 1	CS	:カウント・クロック入力制御
ビット 0	TS	:タイマ動作セット

このレジスタは読出し/書込みが可能です。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CK2	CK1	P4	P3	P2	P1	T2	T1	N/1	R/P	MR2	MR1	0	INT	CS	TS

(オフセット $200 ch0 / $220 ch1 / $240 ch2)

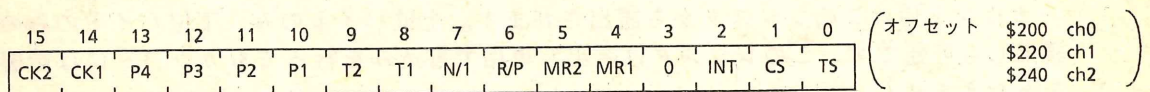



図6.51 タイマ・コントロール・レジスタ

CK2, CK1 : 00－システム・クロック
01－外部クロック
10－他のタイマ出力1(カウンタ0へはカウンタ1の出力、カウンタ1, 2へはカウンタ0の出力)
11－他のタイマ出力2(カウンタ0, 1へはカウンタ2の出力、カウンタ2へはカウンタ1の出力)

(注) 他のタイマ出力1, 2はカスケード接続用です。

P4, P3, P2, P1 : 0001－プリスケーラ2
0010－プリスケーラ4
0011－プリスケーラ8
0100－プリスケーラ16
0101－プリスケーラ32
0110－プリスケーラ64
0111－プリスケーラ128
1xxx－プリスケーラ256

T2, T1 : 00－外部クロックを選択した場合、またはカウント制御機能を使用しない場合に設定する。
10－カウント・スタート機能を設定する。(何度もこの機能を使う場合は一度他の設定にした後、この設定に戻して下さい。)
11－カウント・ウエイト機能を設定する。

N/1 : 0－カウント動作を1度で終了する。(単発)
1－カウント動作を繰り返す。(くり返し)

R/P : 0－出力信号としてパルスを発生する。(パルス)
1－出力レベルを反転する。(レベル反転)

MR2, MR1 : 01－MAX1を使用する。
10－MAX2を使用する。
11－MAX1とMAX2を交互に使用する。(PWM用です。)

INT : 0－割込み要求信号を発生しない。
1－割込み要求信号を発生する。

CS : 0－カウント・クロックをカウンタに入力する。
(カウント動作を開始)
1－カウント・クロックをカウンタに入力しない。
(カウント動作を停止)

TS : 0－カウンタをクリア状態にする。
1－カウンタのクリア状態を解除する。

(注) チャネル0は波形出力機能を持っていないので、チャネル0のコントロール・レジスタにはビット6～4の機能はありません。

第7章 命令セットおよび実行時間

7.1 命令セット

TMP68301のアドレッシング･カテゴリおよび命令セットを次に説明します。

7.1.1 アドレッシング･カテゴリ

実効アドレス･モードはその使用方法によって分類されます。命令の定義の中で次の分類が使われます。

データ　：実効アドレス･モードがデータ･オペランドを参照するために使われる場合、データ参照実効アドレス･モードと考えられます。

メモリ　：実効アドレス･モードがメモリ･オペランドを参照するために使われる場合、メモリ参照実効アドレス･モードと考えられます。

変更可能：実効アドレス･モードが変更可能(書込み可能)なオペランドを参照するために使われる場合、変更可能アドレッシング実効アドレス･モードと考えられます。

制御　：実効アドレス･モードがサイズ指定なしでメモリ･オペランドを参照するために使われる場合、制御アドレッシング実効アドレス･モードと考えられます。

これらのカテゴリを組み合わせて、更に詳細な分類を定義することができます。たとえば、命令の説明では「変更可能メモリ」,「データ変更可能」などを使っています。前者は変更可能であり、メモリ･アドレスでもあるアドレッシング モードを意味し、後者はデータであり変更可能であるアドレッシング･モードを示します。

表7.1に各実効アドレス･モードがどのアドレッシング･カテゴリに属するかを示します。表7.2に命令の要約を示します。

表7.1実効アドレッシング・モードのカテゴリ

実効アドレス･モード	モード	レジスタ	アドレッシング･カテゴリ			
			データ	メモリ	制御	変更可能
Dn	000	レジスタ番号	×	—	—	×
An	001	レジスタ番号	—	—	—	×
(An)	010	レジスタ番号	×	×	×	×
(An) +	011	レジスタ番号	×	×	—	×
- (An)	100	レジスタ番号	×	×	—	×
d16 (An)	101	レジスタ番号	×	×	×	×
d8(An, Xn)	110	レジスタ番号	×	×	×	×
Abs. W	111	000	×	×	×	×
Abs. L	111	001	×	×	×	×
d16(PC)	111	010	×	×	×	—
d8 (PC, Xn)	111	011	×	×	×	—
#xxx	111	100	×	×	—	—

×：属する

表7.2 命令セット (1/5)

ニーモニック	オペレーション	コンディション・コード				
		X	N	Z	V	C
ABCD	(デスティネーション)$_{10}$+(ソース)$_{10}$+X →デスティネーション	*	U	*	U	*
ADD	(デスティネーション)+(ソース) →デスティネーション	*	*	*	*	*
ADDA	(デスティネーション)+(ソース) →デスティネーション	—	—	—	—	—
ADDI	(デスティネーション)+イミディエート・データ →デスティネーション	*	*	*	*	*
ADDQ	(デスティネーション)+イミディエート・データ →デスティネーション	*	*	*	*	*
ADDX	(デスティネーション)+(ソース)+X →デスティネーション	*	*	*	*	*
AND	(デスティネーション)∧(ソース) →デスティネーション	—	*	*	0	0
ANDI	(デスティネーション)∧イミディエート・データ →デスティネーション	—	*	*	0	0
ANDI to CCR	(ソース)∧CCR→CCR	*	*	*	*	*
ANDI to SR	(ソース)∧SR→SR	*	*	*	*	*
ASL, ASR	(デスティネーション)を<カウント>だけ シフト→デスティネーション	*	*	*	*	*
Bcc	条件が真の時、PC+disp→PC	—	—	—	—	—
BCHG	~(デスティネーションの<ビット番号>)→Z ~(デスティネーションの<ビット番号>) →デスティネーションの<ビット番号>	—	—	*	—	—
BCLR	~(デスティネーションの<ビット番号>)→Z 0→デスティネーションの<ビット番号>	—	—	*	—	—
BRA	PC+disp→PC	—	—	—	—	—
BSET	~(デスティネーションの<ビット番号>)→Z 1→デスティネーションの<ビット番号>	—	—	*	—	—
BSR	PC→−(SP);PC+disp→PC	—	—	—	—	—
BTST	~(デスティネーションの<ビット番号>)	—	—	*	—	—

表7.2 命令セット (2/5)

ニーモニック	オペレーション	コンディション・コード				
		X	N	Z	V	C
CHK	Dn > (<ea>) かDn < 0ならトラップ	—	*	U	U	U
CLR	0→デスティネーション	—	0	1	0	0
CMP	(デスティネーション) − (ソース)	—	*	*	*	*
CMPA	(デスティネーション) − (ソース)	—	*	*	*	*
CMPI	(デスティネーション) −イミディエート・データ	—	*	*	*	*
CMPM	(デスティネーション) − (ソース)	—	*	*	*	*
DBcc	コンディションが真でないならDn − 1→Dn Dn≠ − 1ならPC + d16→PC	—	—	—	—	—
DIVS	(デスティネーション) ÷ (ソース) →デスティネーション	—	*	*	*	0
DIVU	(デスティネーション) ÷ (ソース) →デスティネーション	—	*	*	*	0
EOR	(デスティネーション) ⊕ (ソース) →デスティネーション	—	*	*	0	0
EORI	(デスティネーション) ⊕イミディエート・データ →デスティネーション	—	*	*	0	0
EORI to CCR	(ソース) ⊕CCR→CCR	*	*	*	*	*
EORI to SR	(ソース) ⊕SR→SR	*	*	*	*	*
EXG	Xx↔Xy	—	—	—	—	—
EXT	(デスティネーション)の符号を拡張 →デスティネーション	—	*	*	0	0
JMP	デスティネーション→PC	—	—	—	—	—
JSR	PC→ − (SP) ;デスティネーション→PC	—	—	—	—	—
LEA	<ea>→An	—	—	—	—	—
LINK	An→ − (SP) ;SP→An;SP + d16→SP	—	—	—	—	—
LSL, LSR	(デスティネーション)を<カウント>だけ シフト→デスティネーション	*	*	*	0	*

表7.2 命令セット (3/5)

ニーモニック	オペレーション	コンディション・コード				
		X	N	Z	V	C
MOVE	(ソース) →デスティネーション	—	*	*	0	0
MOVE to CCR	(ソース) →CCR	*	*	*	*	*
MOVE to SR	(ソース) →SR	*	*	*	*	*
MOVE from SR	SR→デスティネーション	—	—	—	—	—
MOVE USP	USP→An;An→USP	—	—	—	—	—
MOVEA	(ソース) →デスティネーション	—	—	—	—	—
MOVEM	レジスタ群→デスティネーション (ソース) →レジスタ群	—	—	—	—	—
MOVEP	(ソース) →デスティネーション	—	—	—	—	—
MOVEQ	イミディエート・データ→デスティネーション	—	*	*	0	0
MULS	(デスティネーション) * (ソース) →デスティネーション	—	*	*	0	0
MULU	(デスティネーション) * (ソース) →デスティネーション	—	*	*	0	0
NBCD	0 − (デスティネーション) $_{10}$ − X →デスティネーション	*	U	*	U	*
NEG	0 − (デスティネーション) →デスティネーション	*	*	*	*	*
NEGX	0 − (デスティネーション) − X →デスティネーション	*	*	*	*	*
NOP	—	—	—	—	—	—
NOT	~ (デスティネーション) →デスティネーション	—	*	*	0	0
OR	(デスティネーション) ∨ (ソース) →デスティネーション	—	*	*	0	0
ORI	(デスティネーション) ∨イミディエート・データ →デスティネーション	—	*	*	0	0
ORI to CCR	(ソース) ∨ CCR→CCR	*	*	*	*	*
ORI to SR	(ソース) ∨ SR→SR	*	*	*	*	*

表7.2 命令セット (4/5)

ニーモニック	オペレーション	コンディション・コード				
		X	N	Z	V	C
PEA	<ea>→－(SP)	—	—	—	—	—
RESET	—	—	—	—	—	—
ROL,ROR	(デスティネーション)を<カウント>だけ回転 →デスティネーション	—	*	*	0	*
ROXL, ROXR	(デスティネーション)を<カウント>だけ回転 →デスティネーション	*	*	*	0	*
RTE	(SP)＋→SR; (SP)＋→PC	*	*	*	*	*
RTR	(SP)＋→CC; (SP)＋→PC	*	*	*	*	*
RTS	(SP)＋→PC	—	—	—	—	—
SBCD	(デスティネーション)$_{10}$－(ソース)$_{10}$－X →デスティネーション	*	U	*	U	*
Scc	コンディションが真の時は すべて1($FF) →デスティネーション その他は すべて0($00)→デスティネーション	—	—	—	—	—
STOP	イミディエート・データ→SR;STOP	*	*	*	*	*
SUB	(デスティネーション)－(ソース) →デスティネーション	*	*	*	*	*
SUBA	(デスティネーション)－(ソース) →デスティネーション	—	—	—	—	—
SUBI	(デスティネーション)－イミディエート・データ →デスティネーション	*	*	*	*	*
SUBQ	(デスティネーション)－イミディエート・データ →デスティネーション	*	*	*	*	*
SUBX	(デスティネーション)－(ソース)－X →デスティネーション	*	*	*	*	*
SWAP	レジスタ[31:16]↔レジスタ[15:0]	—	*	*	0	0
TAS	(デスティネーション)テスト→CC; 1→デスティネーションのビット7	—	*	*	0	0

表7.2 命令セット (5/5)

ニーモニック	オペレーション	コンディション・コード				
		X	N	Z	V	C
TRAP	PC→－(SSP);SR→－(SSP);(ベクタ)→PC	―	―	―	―	―
TRAPV	Vフラグがセットしていればトラップ	―	―	―	―	―
TST	(デスティネーション)テスト→CC	―	*	*	0	0
UNLK	An→SP; (SP)＋→An	―	―	―	―	―

→ ： 左のオペランドが右のオペランドで指定される位置に転送される。
↔ ： 2つのオペランド内容が交換される。
＋ ： 両オペランド内容が加算される。
－ ： 右側のオペランドが左側のオペランドから減算される。
* ： 左右のオペランドが乗算される
/ ： 左のオペランドが右のオペランドで除算される。
∧ ： 左右オペランドは理論的にANDされます。
∨ ： 左右オペランドは理論的にORされます。
⊕ ： 左右オペランドは理論的にEORされます。
< ： 関係テスト、左のオペランドが右のオペランドより小なら真。
> ： 関係テスト、左のオペランドが右のオペランドより大きければ真。
~ ： オペランドは理論的に反転されます。
[ ] ： ビット番号

* ： 操作の結果によりセットされる。
－ ： 操作によって変化しない。
0 ： クリアされる。
1 ： セットされる。
U ： 操作のあと不定となる。

7.1.2 命令プリフェッチ

TMP68301は2ワードの命令プリフェッチ機構を使って、性能向上を図っています。この機能は、関係するマイクロコードの動作によって記述されます。命令の実行が、その命令用のマイクロ･ルーチンに入った時開始されるように定義されている場合、プリフェッチ機構の次のような特徴が現れます。

1) 命令の実行開始時、オペランド･ワードとその次のワードが既にフェッチされていて、オペレーション･ワードは、命令デコーダの中に入っています。
2) 複数ワードの命令の場合、命令を構成する各ワードが内部的に使われるたびに次のワードが命令ストリームからフェッチされて次々に置き換えられます。
3) オペレーション･ワードが捨てられて、次の命令デコードが開始される時、命令ストリームからの最後のフェッチが行われます。
4) 分岐を起こすシングル･ワード命令の場合、第2ワード(プリフェッチ機構内の)は使われません。しかし、このワードは先行する命令によってフェッチされるのでこの余分なフェッチを避けることは不可能です。
5) 割込みまたはトレースの例外事象が発生した場合、プリフェッチされたワードは両方共使われません。
6) プログラム･カウンタは通常、命令ストリームから最近フェッチされたワードを示しています。

7.2 命令実行時間

ここでは命令の実行時間を外部クロック(CLK)の単位で示します。このタイミング･データにおいて、メモリ読出しおよび書込みのサイクル･タイムは両方とも4クロックであると仮定しています。これより長い場合は、その分だけ加える必要があります。タイミングのデータと一緒に各命令のバス読出しおよび書込みサイクルの回数を示します。この値は実行周期の後に(r/w)の形(R=読出しサイクルの回数、W=書込みサイクルの回数)で示されています。

(注)

周期の数は命令フェッチおよび適用可能なオペランド･フェッチおよび格納をすべて含みます。

7.2.1 オペランドの実効アドレスの計算時間

命令の実効アドレスを計算するのに必要なクロック数を表7.3に示します。この中には拡張ワードのフェッチ,アドレスの計算,およびメモリ･オペランドのフェッチに必要な時間が含まれています。実効アドレスの計算には書込みサイクルは含まれないことに注意してください。

表7.3 実効アドレスの計算時間

	アドレッシング・モード	バイト，ワード	ロング・ワード
Dn An	レジスタ データ・レジスタ直接 アドレス・レジスタ直接	0 (0/0) 0 (0/0)	0 (0/0) 0 (0/0)
(An) (An) +	メモリ アドレス・レジスタ間接 アドレス・レジスタ間接(ポストインクリメント付き)	4 (1/0) 4 (1/0)	8 (2/0) 8 (2/0)
− (An) d16 (PC)	アドレス・レシスタ間接(プリデクリメント付き) アドレス・レシスタ間接(ディスプレースメント付き)	6 (1/0) 8 (2/0)	10 (2/0) 12 (3/0)
d8 (An, Xn) * Abs. W	アドレス・レジスタ間接(インデックス付き) 絶対ショート	10 (2/0) 8 (2/0)	14 (3/0) 12 (3/0)
Abs. L d16 (PC)	絶対ロング ディスプレースメント付きPC	12 (3/0) 8 (2/0)	16 (4/0) 12 (3/0)
d8 (PC, Xn) * #xxx	インデックス付きPC イミディエート	10 (2/0) 4 (1/0)	14 (3/0) 8 (2/0)

＊：インデックス レジスタ (Xn) のサイズは実行時間には影響しません。

7.2.2 MOVE命令の実行時間

MOVE命令のクロック数を表7.4および7.5に示します。この値は命令フェッチ・オペランドの読出し、およびオペランドの書込みを含んでいます。

表7.4 バイトおよびワードのMOVE命令の実行時間

ソース	デスティネーション								
	Dn	An	(An)	(An) +	− (An)	d16(An)	d8(An,Xn)*	Abs.W	Abs.L
Dn	4 (1/0)	4 (1/0)	8 (1/1)	8 (1/1)	8 (1/1)	12 (2/1)	14 (2/1)	12 (2/1)	16 (3/1)
An	4 (1/0)	4 (1/0)	8 (1/1)	8 (1/1)	8 (1/1)	12 (2/1)	14 (2/1)	12 (2/1)	16 (3/1)
(An)	8 (2/0)	8 (2/0)	12 (2/1)	12 (2/1)	12 (2/1)	16 (3/1)	18 (3/1)	16 (3/1)	20 (4/1)
(An) +	8 (2/0)	8 (2/0)	12 (2/1)	12 (2/1)	12 (2/1)	16 (3/1)	18 (3/1)	16 (3/1)	20 (4/1)
− (An)	10 (2/0)	10 (2/0)	14 (2/1)	14 (2/1)	14 (2/1)	18 (3/1)	20 (3/1)	18 (3/1)	22 (4/1)
d16 (An)	12 (3/0)	12 (3/0)	16 (3/1)	16 (3/1)	16 (3/1)	20 (4/1)	22 (4/1)	20 (4/1)	24 (5/1)
d8 (An, Xn)*	14 (3/0)	14 (3/0)	18 (3/1)	18 (3/1)	18 (3/1)	22 (4/1)	24 (4/1)	22 (4/1)	26 (5/1)
Abs. W	12 (3/0)	12 (3/0)	16 (3/1)	16 (3/1)	16 (3/1)	20 (4/1)	22 (4/1)	20 (4/1)	24 (5/1)
Abs. L	16 (4/0)	16 (4/0)	20 (4/1)	20 (4/1)	20 (4/1)	24 (5/1)	26 (5/1)	24 (5/1)	28 (6/1)
d16 (PC)	12 (3/0)	12 (3/0)	16 (3/1)	16 (3/1)	16 (3/1)	20 (4/1)	22 (4/1)	20 (4/1)	24 (5/1)
d8 (PC, Xn) *	14 (3/0)	14 (3/0)	18 (3/1)	18 (3/1)	18 (3/1)	22 (4/1)	24 (4/1)	22 (4/1)	26 (5/1)
#xxx	8 (2/0)	8 (2/0)	12 (2/1)	12 (2/1)	12 (2/1)	16 (3/1)	18 (3/1)	16 (3/1)	20 (4/1)

＊：インデックス・レジスタ (Xn) のサイズは実行時間には影響しません。

表7.5 ロング・ワードのMOVE命令の実行時間

ソース	デスティネーション								
	Dn	An	(An)	(An)+	−(An)	d16(An)	d8(An,Xn)*	Abs.W	Abs.L
Dn	4 (1/0)	4 (1/0)	12 (1/2)	12 (1/2)	12 (1/2)	16 (2/2)	18 (2/2)	16 (2/2)	20 (3/2)
An	4 (1/0)	4 (1/0)	12 (1/2)	12 (1/2)	12 (1/2)	16 (2/2)	18 (2/2)	16 (2/2)	20 (3/2)
(An)	12 (3/0)	12 (3/0)	20 (3/2)	20 (3/2)	20 (3/2)	24 (4/2)	26 (4/2)	24 (4/2)	28 (5/2)
(An) +	12 (3/0)	12 (3/0)	20 (3/2)	20 (3/2)	20 (3/2)	24 (4/2)	26 (4/2)	24 (4/2)	28 (5/2)
− (An)	14 (3/0)	14 (3/0)	22 (3/2)	22 (3/2)	22 (3/2)	26 (4/2)	28 (4/2)	26 (4/2)	30 (5/2)
d16 (An)	16 (4/0)	16 (4/0)	24 (4/2)	24 (4/2)	24 (4/2)	28 (5/2)	30 (5/2)	28 (5/2)	32 (6/2)
d8 (An, Xn) *	18 (4/0)	18 (4/0)	26 (4/2)	26 (4/2)	26 (4/2)	30 (5/2)	32 (5/2)	30 (5/2)	34 (6/2)
Abs. W	16 (4/0)	16 (4/0)	24 (4/2)	24 (4/2)	24 (4/2)	28 (5/2)	30 (5/2)	28 (5/2)	32 (6/2)
Abs. L	20 (5/0)	20 (5/0)	28 (5/2)	28 (5/2)	28 (5/2)	32 (6/2)	34 (6/2)	32 (6/2)	36 (7/2)
d16 (PC)	16 (4/0)	16 (4/0)	24 (4/1)	24 (4/2)	24 (4/2)	28 (5/2)	30 (5/2)	28 (5/2)	32 (5/2)
d8 (PC, Xn) *	18 (4/0)	18 (4/0)	26 (4/2)	26 (4/2)	26 (4/2)	30 (5/2)	32 (5/2)	30 (5/2)	34 (6/2)
#xxx	12 (3/0)	12 (3/0)	20 (3/2)	20 (3/2)	20 (3/2)	24 (4/2)	26 (4/2)	24 (4/2)	28 (5/2)

* : インデックス・レジスタ (Xn) のサイズは実行時間には影響しません。

7.2.3 標準命令の実行時間

表7.6に演算の実行，結果の格納，および次の命令の読出しに必要な時間を示します。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。実効アドレスの計算が必要な場合は、実効アドレスの計算に要するクロック数とバス読出しおよび書込みサイクルの回数を加算して下さい。

表7.6で各記号の意味は次の通りです。

An＝アドレス・レジスタ・オペランド
Dn＝データ・レジスタ・オペランド
ea＝実効アドレスによって示されるオペランド
M＝メモリ実効アドレス・オペランド

表7.6 標準命令の実行時間

命　令	サイズ	op<ea>, An^	op<ea>, Dn	opDn, <M>
ADD	バイト，ワード	8 (1/0) +	4 (1/0) +	8 (1/1) +
	ロング・ワード	6 (1/0) + **	6 (1/0) + **	12 (1/2) +
AND	バイト，ワード	—	4 (1/0) +	8 (1/1) +
	ロング・ワード	—	6 (1/0) + **	12 (1/2) +
CMP	バイト，ワード	6 (1/0) +	4 (1/0) +	—
	ロング・ワード	6 (1/0) +	6 (1/0) +	—
DIVS	—	—	158 (1/0) + *	—
DIVU	—	—	140 (1/0) + *	—
EOR	バイト，ワード	—	4 (1/0) ***	8 (1/1) +
	ロング・ワード	—	8 (1/0) ***	12 (1/2) +
MULS	—	—	70 (1/0) + *	—
MULU	—	—	70 (1/0) + *	—
OR	バイト，ワード	—	4 (1/0) +	8 (1/1) +
	ロング・ワード	—	6 (1/0) + **	12 (1/2) +
SUB	バイト，ワード	8(1/0) +	4 (1/0) +	8 (1/1) +
	ロング・ワード	6(1/0) + **	6 (1/0) + **	12 (1/2) +

+　：実効アドレスの計算時間を加算する。
^　：ワードとロング・ワードのみ。
*　：最大値。
**　：実効アドレス・モードが「レジスタ直接」か「イミディエート」の場合，命令全体の合計が8クロック。
***　：実効アドレス・モードは「データ・レジスタ直接」のみ。

DIVS, DIVU - TMP68301の除算アルゴリズムでは、タイミングの最大値と最小値の差は10%以下です。

MULS, MULU- 乗算アルゴリズムは38＋2nクロックを要します。ここでnは次のように定義されます。

MULU ：n＝<ea>中の1の数。
MULS ：n＝<ea>のLSBに0を付加した17ビットのソースにおける10または01パターンの数。最大値は、ソースが$5555の時。

7.2.4 イミディエート命令の実行時間

表7.7クロック数には、イミディエート・オペランドのフェッチ，オペレーションの実行，結果の格納，および次の命令の読出しに必要な時間が含まれています。実効アドレスの計算が必要な場合は、実効アドレスの計算に要するクロック数とバス読出しおよび書込みサイクルの回数を加算して下さい。

表7.7で各記号の意味は次の通りです。

#　＝イミディエート・オペランド
Dn　＝データ・レジスタ・オペランド
An　＝アドレス・レジスタ・オペランド
M　＝メモリ・オペランド
SR　＝ステータス・レジスタ

表7.7 イミディエート命令の実行時間

命　令	サイズ	op #, Dn	op #, An	op #, M
ADDI	バイト，ワード	8 (2/0)	—	12 (2/1) +
	ロング・ワード	16 (3/0)	—	20 (3/2) +
ADDQ	バイト，ワード	4 (1/0)	8 (1/0) *	8 (1/1) +
	ロング・ワード	8 (1/0)	8 (1/0)	12 (1/2) +
ANDI	バイト，ワード	8 (2/0)	—	12 (2/1) +
	ロング・ワード	16 (3/0)	—	20 (3/1) +
CMPI	バイト，ワード	8 (2/0)	—	8 (2/0) +
	ロング・ワード	14 (3/0)	—	12 (3/0) +
EORI	バイト，ワード	8 (2/0)	—	12 (2/1) +
	ロング・ワード	16 (3/0)	—	20 (3/2) +
MOVEQ	ロング・ワード	4 (1/0)	—	—
ORI	バイト，ワード	8 (2/0)	—	12 (2/1) +
	ロング・ワード	16 (3/0)	—	20 (3/2) +
SUBI	バイト，ワード	8 (2/0)	—	12 (2/1) +
	ロング・ワード	16 (3/0)	—	20 (3/2) +
SUBQ	バイト，ワード	4 (1/0)	8 (1/0) *	8 (1/1) +
	ロング・ワード	8 (1/0)	8 (1/0)	12 (1/2) +

+ ：実効アドレスの計算時間を加算する。
* ：ワードのみ

7.2.5 単一オペランド命令の実行時間

表7.8に単一オペランド命令に要するクロック数を示します。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。実効アドレスの計算が必要な場合は、実効アドレスの計算に要するクロック数とバス読出しおよび書込みサイクルの回数を加算して下さい。

表7.8 単一オペランド命令の実行時間

命　令	サイズ	レジスタ	メモリ
CLR	バイト, ワード	4 (1/0)	8 (1/1) +
	ロング・ワード	6 (1/0)	12 (1/2) +
NBCD	バイト	6 (1/0)	8 (1/1) +
NEG	バイト, ワード	4 (1/0)	8 (1/1) +
	ロング・ワード	6 (1/0)	12 (1/2) +
NEGX	バイト, ワード	4 (1/0)	8 (1/1) +
	ロング・ワード	6 (1/0)	12 (1/2) +
NOT	バイト, ワード	4 (1/0)	8 (1/1) +
	ロング・ワード	6 (1/0)	12 (1/2) +
Scc	バイト, 偽	4 (1/0)	8 (1/1) +
	バイト, 真	6 (1/0)	8 (1/1) +
TAS	バイト	4 (1/0)	10 (1/1) +
TST	バイト, ワード	4 (1/0)	4 (1/0) +
	ロング・ワード	4 (1/0)	4 (1/0) +

＋ ： 実効アドレスの計算時間を加算する必要あり。

7.2.6 シフト/ローテイト命令の実行時間

表7.9にシフトとローテイト命令に要するクロック数を示します。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。実効アドレスの計算が必要な場合は、実効アドレスの計算に要するクロック数とバス読出しおよび書込みサイクルの回数を加算して下さい。

表7.9 シフト/ローテイト命令の実行時間

命　令	サイズ	レジスタ	メモリ
ASR, ASL	バイト, ワード	6 + 2n (1/0)	8 (1/1) +
	ロング・ワード	8 + 2n (1/0)	—
LSR, LSL	バイト, ワード	6 + 2n (1/0)	8 (1/1) +
	ロング・ワード	8 + 2n (1/0)	—
ROR, ROL	バイト, ワード	6 + 2n (1/0)	8 (1/1) +
	ロング・ワード	8 + 2n (1/0)	—
ROXR, ROXL	バイト, ワード	6 + 2n (1/0)	8 (1/1) +
	ロング・ワード	8 + 2n (1/0)	—

＋ ： 実効アドレスの計算時間を加算する。
n ： シフトまたはローテイトの回数。

7.2.7 ビット操作命令の実行時間

表7.10にビット操作命令に要するクロック数を示します。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。実効アドレスの計算が必要な場合は、実効アドレスの計算に要するクロック数とバス読出しおよび書込みサイクルの回数を加算して下さい。

表7.10 ビット操作命令の実行時間

命令	サイズ	ダイナミック		スタティック	
		レジスタ	メモリ	レジスタ	メモリ
BCHG	バイト	—	8 (1/1) +	—	12 (2/1) +
	ロング・ワード	8 (1/0) *	—	12 (2/0) *	—
BCLR	バイト	—	8 (1/1) +	—	12 (2/1) +
	ロング・ワード	10 (1/0) *	—	14 (2/0) *	—
BSET	バイト	—	8 (1/1) +	—	12 (2/1) +
	ロング ワード	8 (1/0) *	—	12 (2/0) *	—
BTST	バイト	—	4 (1/0) +	—	8 (2/0) +
	ロング・ワード	6 (1/0)	—	10 (2/0)	—

+ : 実効アドレスの計算時間を加算する。
* : 最大値

7.2.8 コンディション付き命令の実行時間

表7.11にコンディション付き命令に要するクロック数を示します。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。実効アドレスの計算が必要な場合は、実効アドレスの計算に要するクロック数とバス読出しおよび書込みサイクルの回数を加算して下さい。

表7.11 コンディション付き命令の実行時間

命令	ディスプレースメント	分岐が起る	分岐が起らない
Bcc	バイト	10 (2/0)	8 (1/0)
	ワード	10 (2/0)	12 (2/0)
BRA	バイト	10 (2/0)	—
	ワード	10 (2/0)	—
BSR	バイト	18 (2/2)	—
	ワード	18 (2/2)	—
DBcc	CC真	—	12 (2/0)
	CC偽	10 (2/0)	14 (3/0)

7.2.9 JMP, JSR, LEA, PEA, および MOVEM命令の実行時間

表7.12にJMP, JSR, LEA, PEA, および MOVEM命令に要するクロック数を示します。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。

表7.12 JMP, JSR, LEA, PEA, および MOVEM命令の実行時間

命令	サイズ	(An)	(An)+	−(An)	d16 (An)	d8 (An, Xn) *	Abs.W	Abs.L	d16 (PC)	d8 (PC, Xn) *
JMP	—	8 (2/0)	—	—	10 (2/0)	14 (3/0)	10 (2/0)	12 (3/0)	10 (2/0)	14 (3/0)
JSR	—	16 (2/2)	—	—	18 (2/2)	22 (2/2)	18 (2/2)	20 (3/2)	18 (2/2)	22 (2/2)
LEA	—	4 (1/0)	—	—	8 (2/0)	12 (2/0)	8 (2/2)	12 (3/0)	8 (2/0)	12 (2/0)
PEA	—	12 (1/2)	—	—	16 (2/2)	20 (2/2)	16 (2/2)	20 (3/2)	16 (2/2)	20 (2/2)
MOVEM M→R	ワード	12 + 4n (3 + n/0)	12 + 4n (3 + n/0)	—	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)	16 + 4n (4 + 2n/0)	20 + 4n (5 + n/0)	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)
	ロング・ワード	12 + 8n (3 + 2n/0)	12 + 8n (3 + 2n/0)	—	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)	16 + 8n (4 + 2n/0)	20 + 8n (5 + 2n/0)	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)
MOVEM R→M	ワード	8 + 4n (2/n)	—	8 + 4n (2/n)	12 + 4n (3/n)	14 + 4n (3/n)	12 + 4n (3/n)	16 + 4n (4/n)	—	—
	ロング・ワード	8 + 8n (2/2n)	—	8 + 8n (2/2n)	12 + 8n (3/2n)	14 + 8n (3/2n)	12 + 8n (3/2n)	16 + 8n (4/2n)	—	—

n : 転送するレジスタの数

* : インデックス レジスタ (Xn) のサイズは実行時間には影響しません。

7.2.10 多倍精度命令の実行時間

表7.13に多倍精度命令に要するクロック数を示します。表7.13のクロック数には両オペランドのフェッチ，オペレーションの実行，結果の格納，次命令の読出しに必要な時間が含まれています。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。

表7.13で各記号の意味は次の通りです。

Dn ＝データ・レジスタ・オペランド

M ＝メモリ・オペランド

表7.13 多倍精度命令の実行時間

命令	サイズ	op Dn, Dn	op M, M
ADDX	バイト，ワード	4 (1/0)	18 (3/1)
	ロング・ワード	8 (1/0)	30 (5/2)
CMPM	バイト，ワード	—	12 (3/0)
	ロング・ワード	—	20 (5/0)
SUBX	バイト，ワード	4 (1/0)	18 (3/1)
	ロング・ワード	8 (1/0)	30 (5/2)
ABCD	バイト	6 (1/0)	18 (3/1)
SBCD	バイト	6 (1/0)	18 (3/1)

7.2.11 その他の命令の実行時間

表7.14と表7.15その他の命令に要するクロック数を示します。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。実効アドレスの計算が必要な場合は、実効アドレスの計算に要するクロックの回数とバス読出しおよび書込みサイクル数を加算して下さい。

表7.14 その他の命令の実行時間

命　令	サイズ	レジスタ	メモリ
ANDI to CCR	バイト	20 (3/0)	−
ANDI to SR	ワード	20 (3/0)	−
CHK	−	10 (1/0) +	−
EORI to CCR	バイト	20 (3/0)	−
EORI to SR	ワード	20 (3/0)	−
ORI to CCR	バイト	20 (3/0)	−
ORI to SR	ワード	20 (3/0)	−
MOVE from SR	−	6 (1/0)	8 (1/1) +
MOVE to CCR	−	12 (1/0)	12 (1/0) +
MOVE to SR	−	12 (1/0)	12 (1/0) +
EXG	−	6 (1/0)	−
EXT	ワード	4 (1/0)	−
	ロング・ワード	4 (1/0)	−
LINK	−	16 (2/2)	−
MOVE from USP	−	4 (1/0)	−
MOVE to USP	−	4 (1/0)	−
NOP	−	4 (1/0)	−
RESET	−	132 (1/0)	−
RTE	−	20 (5/0)	−
RTR	−	20 (5/0)	−
RTS	−	16 (4/0)	−
STOP	−	4 (0/0)	−
SWAP	−	4 (1/0)	−
TRAPV (トラップなし)	−	4 (1/0)	−
UNLK	−	12 (3/0)	−

+ : 実効アドレスの計算時間を加算する。

表7.15 MOVEP命令の実行時間

命　令	サイズ	レジスタ→メモリ	メモリ→レジスタ
MOVEP	ワード	16 (2/2)	16 (4/0)
	ロング・ワード	24 (2/4)	24 (6/0)

7.2.12 例外処理の実行時間

表7.16に例外処理のクロック数を示します。表7.16のクロック数には全てのスタック操作，ベクタのフェッチ，および例外処理ルーチンの最初の命令のフェッチが含まれています。バス読出しおよび書込みサイクルの回数は括弧の中に(r/w)で示します。

表7.16 例外処理の実行時間

例　外	クロック数
アドレス・エラー	50 (4/7)
バス・エラー	50 (4/7)
CHK命令(トラップあり)	44 (5/3) +
0による除算	42 (5/3)
イリーガル命令	34 (4/3)
割込み	44 (5/3)*
特権違反	34 (4/3)
リセット**	40 (6/0)
トレース	34 (4/3)
TRAP命令	38 (4/3)
TRAPV命令(トラップあり)	34 (4/3)

第8章 電気的特性

この章では、TMP68301の電気的特性およびタイミングを説明します。

8.1 最大定格

項目	記号	定格値	単位
		TMP68301	
電源電圧	Vcc	-0.3～+6.5	V
入力電圧	Vin	-0.3～+6.5	V
動作温度	Ta	0～+70	℃
保存温度	Tstg	-55～+150	℃

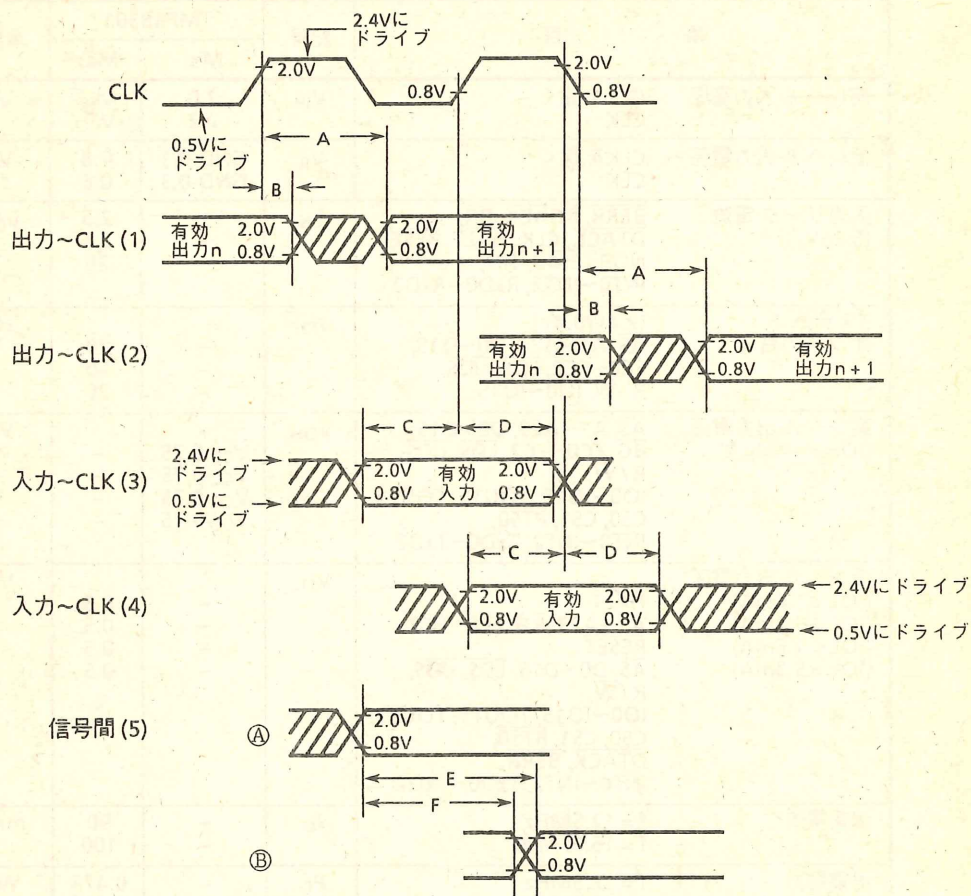
このデバイスには、高圧静電気や電界による破損に備えて、入力保護回路が付いていますが、最大定格電圧より高い電圧をかけることは避けて下さい。使用していない入力ピンをGNDまたはV_{CC}につないで下さい。

8.2 DC特性

(VCC = 5.0V ± 5%, GND = 0V, Ta = 0~70°C)

項目		記号	TMP68301		単位
			Min	Max	
高レベル入力電圧	CLKを除く CLK	V_{IH}	2.0 2.2	V_{CC} V_{CC}	V
低レベル入力電圧	CLKを除く CLK	V_{IL}	GND-0.3 GND-0.3	0.8 0.6	V
入力リーク電流 (5.25V)	BERR, BGACK, BR, DTACK, CLK, HALT, RESET, NOR / EMU, TIN, INT0~INT2, RxD0~RxD2	I_{IN}	– – –	2.5 2.5 20	μA
3ステート・ オフ入力電流	(2.4V/0.4V) AS, A1~A23, D0~D15, FC0~FC2, LDS, UDS, R / W, IO0~IO15	I_{TSI}	 – – –	 20 20 20	μA
高レベル出力電圧 (IOH = -400μA)	AS, A1~A23, D0~D15, BG, FC0~FC2, LDS, UDS, R / W, IO0~IO15, TOUT1, TOUT2, CS0, CS1, RTS0, INT0~INT2, TxD0~TxD2	V_{OH}	– V_{CC}-0.75 V_{CC}-0.75 V_{CC}-0.75 V_{CC}-0.75	 – – – –	V
低レベル出力電圧 (IOL = 1.6mA) (IOL = 3.2mA) (IOL = 1.6mA) (IOL = 5.3mA)	 HALT A1~A23, BG, FC0~FC2 RESET AS, D0~D15, LDS, UDS, R / W, IO0~IO15, TOUT1, TOUT2, CS0, CS1, RTS0, DTACK, BERR, INT0~INT2, TxD0~TxD2	V_{OL}	 – – – –	 0.5 0.5 0.5 0.5	V
消費電流	f = 12.5MHz f = 16.67MHz	I_D	– –	90 100	mA
消費電力	f = 12.5MHz f = 16.67MHz	P_D	– –	0.473 0.525	W
入力容量 (Vin = 0V, Ta = 25℃ : 周波数 = 1MHz)*		C_{IN}	–	20.0	pF
負荷容量	HALT その他のピン	C_L	– –	70 130	pF

*： 入力容量は、抜き取りテストによる値です。

AC電気的特性の定義

A：最大出力ディレイ

B：最小出力ホールド時間

C：最小入力セットアップ時間

D：最小入力ホールド時間

E：信号Ⓐ有効~信号Ⓑ有効時間

F：信号Ⓐ有効~信号Ⓑ無効時間

8.3 AC特性－クロック・タイミング

(V$_{CC}$ = 5.0V ± 5%, GND = 0V, Ta = 0~70℃; 図8.1参照)

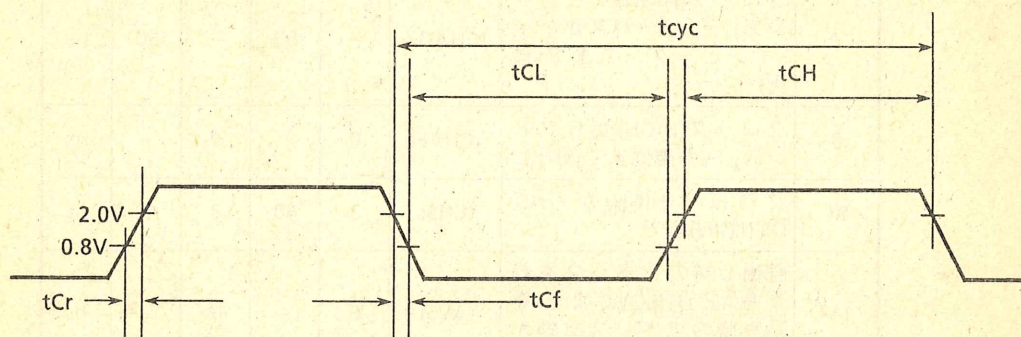
項　目	記号	12.5MHz		16.67MHz		単位
		最小	最大	最小	最大	
動作周波数	f	4.0	12.5	8.0	16.67	MHz
周期	tcyc	80	250	60	125	ns
クロック・パルス幅	tCL tCH	35 35	125 125	27 27	62.5 62.5	ns
立上りおよび立下り時間	tCr tCf	– –	5 5	– –	5 5	ns

(注)　特に指定がない場合, 高レベル電圧は2.0[V], 低レベル電圧は0.8[V]としてタイミングを測定してあります。電圧波形は、この範囲の外側から始まり、この範囲内で直線的に変化しなければなりません。

図8.1　クロック入力タイミング

8.4 AC特性－読出し/書込みサイクル (1/4)

(V_{CC} = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; 図8.2, 8.3参照)

番号	項　目	記号	12.5MHz		16.67MHz		単位
			最小	最大	最小	最大	
1	クロック周期	tCYC	80	250	60	125	ns
2	クロック幅(LOW)	tCL	35	125	27	62.5	ns
3	クロック幅(HIGH)	tCH	35	125	27	62.5	ns
4	クロック立下り時間	tCf	–	5	–	5	ns
5	クロック立上り時間	tCr	–	5	–	5	ns
6	クロック(LOW)からアドレス有効まで	tCLAV	–	50	–	50	ns
6A	クロック(HIGH)からFC有効まで	tCHFCV	–	45	–	45	ns
7	クロック(HIGH)からアドレス, データ・バスがハイ・インピーダンスまで(最大)	tCHADZ	–	60	–	50	ns
8	クロック(HIGH)からアドレス, FCが無効まで(最小)	tCHAFI	0	–	0	–	ns
9[1]	クロック(HIGH)から$\overline{AS}$, $\overline{DS}$ (LOW)まで	tCHSL	3	40	3	40	ns
11[2]	読出し時のアドレス有効から$\overline{AS}$, $\overline{DS}$(LOW)まで/書込み時のアドレス有効から$\overline{AS}$ (LOW)まで	tAVSL	0	–	15	–	ns
11A[2]	読出し時のFC有効から$\overline{AS}$, $\overline{DS}$ (LOW)まで/書込み時のFC有効から$\overline{AS}$ (LOW)まで	tFCVSL	60	–	30	–	ns
12[1]	クロック(LOW)から$\overline{AS}$, $\overline{DS}$ (HIGH)まで	tCLSH	–	40	–	40	ns
13[2]	$\overline{AS}$, $\overline{DS}$ (HIGH)アドレス/FC無効まで	tSHAFI	20	–	10	–	ns
14[2]	読出し時の$\overline{AS}$, $\overline{DS}$幅(LOW)/書込み時の$\overline{AS}$幅(LOW)	tSL	160	–	120	–	ns
14A[2]	書込み時の$\overline{DS}$幅(LOW)	tDSL	80	–	60	–	ns
15[2]	$\overline{AS}$, $\overline{DS}$幅(HIGH)	tSH	65	–	60	–	ns

8.4 AC特性－読出し/書込みサイクル (2/4)

(V_{CC} = 5.0V ± 5%, GND = 0V, Ta = 0～70°C; 図8.2, 8.3参照)

番号	項　目	記号	12.5MHz		16.67MHz		単位
			最小	最大	最小	最大	
16	クロック(HIGH)からコントロール・バスがハイ・インピーダンスまで	tCHCZ	–	60	–	50	ns
17[2]	読出し時の$\overline{AS}$, $\overline{DS}$ (HIGH)からR/$\overline{W}$(HIGH)まで	tSHRH	20	–	10	–	ns
18[1]	クロック(HIGH)からR/$\overline{W}$(HIGH)まで	tCHRH	0	40	0	40	ns
20[1]	書込み時のクロックからR/$\overline{W}$(LOW)まで	tCHRL	0	40	0	40	ns
20A[2,6]	書込み時の$\overline{AS}$(LOW)からR/$\overline{W}$有効まで	tASRV	–	10	–	10	ns
21[2]	書込み時のアドレス有効からR/$\overline{W}$(LOW)まで書込み時のFC有効からR/$\overline{W}$(LOW)まで	tAVRL	0	–	0	–	ns
21A[2]	書込み時のFC有効からR/$\overline{W}$(LOW)まで	tFCVRL	30	–	20	–	ns
22[2]	書込み時のR/$\overline{W}$ (LOW)から$\overline{DS}$(LOW)まで	tRLSL	30	–	20	–	ns
23	書込み時のクロック(LOW)から出力データ有効まで	tCLDO	–	50	–	50	ns
25[2]	書込み時の$\overline{AS}$, $\overline{DS}$(HIGH)から出力データ無効まで	tSHDOI	20	–	15	–	ns
26[2]	書込み時の出力データ有効から$\overline{DS}$(LOW)まで	tDOSL	20	–	15	–	ns
27[5]	読出時のクロック(LOW)に対する入力データのセットアップ時間	tDICL	10	–	7	–	ns
28[2]	$\overline{AS}$, $\overline{DS}$ (HIGH)から$\overline{DTACK}$(HIGH)まで	tSHDAH	0	150	0	110	ns
29	$\overline{AS}$, $\overline{DS}$のネゲートからデータが無効になるまで(リード時のホールド時間)	tSHDII	0	–	0	–	ns

8.4 AC特性－読出し/書込みサイクル (3/4)

(V_{CC} = 5.0V ± 5%, GND = 0V, Ta = 0～70°C; 図8.2, 8.3参照)

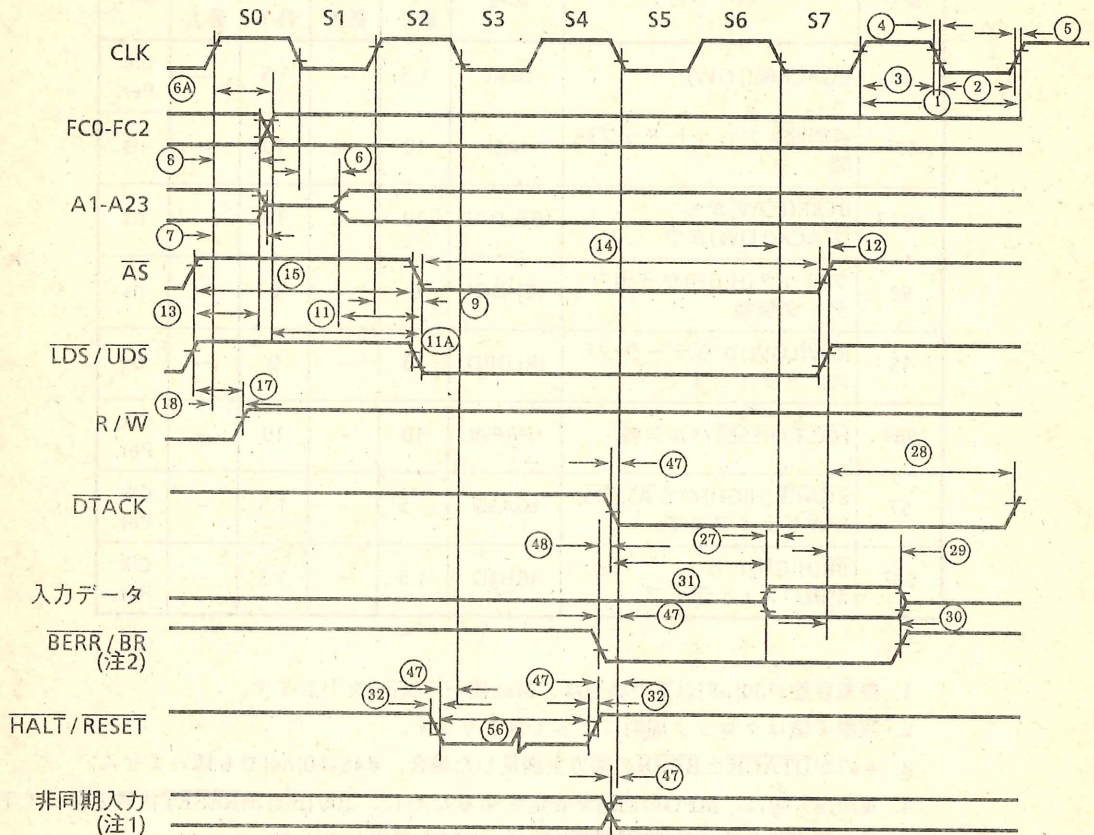
番号	項目	記号	12.5MHz		16.67MHz		単位
			最小	最大	最小	最大	
30	$\overline{AS}$, $\overline{DS}$ (HIGH)から$\overline{BERR}$ (HIGH)まで	tSHBEH	0	–	0	–	ns
31[2,5]	入力データに対する$\overline{DTACK}$(LOW)のセットアップ時間	tDALDI	–	50	–	40	ns
32	$\overline{HALT}$と$\overline{RESET}$入力の遷移時間	tRHr, f	0	200	0	150	ns
33	クロック(HIGH)から$\overline{BG}$(LOW)まで	tCHGL	–	40	–	40	ns
34	クロック(HIGH)から$\overline{BG}$(HIGH)まで	tCHGH	–	40	–	40	ns
35	$\overline{BR}$(LOW)から$\overline{BG}$(LOW)まで	tBRLGL	1.5	3.5	1.5	3.5	Clk. Per.
36[7]	$\overline{BR}$(HIGH)から$\overline{BG}$(HIGH)まで	tBRHGH	1.5	3.5	1.5	3.5	Clk. Per.
37	$\overline{BGACK}$(LOW)から$\overline{BG}$(LOW)まで	tGALGH	1.5	3.5	1.5	3.5	Clk. Per.
37A[8]	$\overline{BGACK}$(LOW)から$\overline{BR}$(HIGH)まで	tGALBRH	20	1.5 Clocks	10	1.5 Clocks	ns
38	$\overline{AS}$(HIGH)の時, $\overline{BG}$(LOW)から制御, アドレス, データ・バス, ハイ・インピーダンスまで	tGLZ	–	60	–	50	ns
39	$\overline{BG}$幅(HIGH)	tGH	1.5	–	1.5	–	Clk. Per.

8.4 AC特性－読出し/書込みサイクル (4/4)

(V_{CC} = 5.0V ± 5%, GND = 0V, Ta = 0～70℃; 図8.2, 8.3参照)

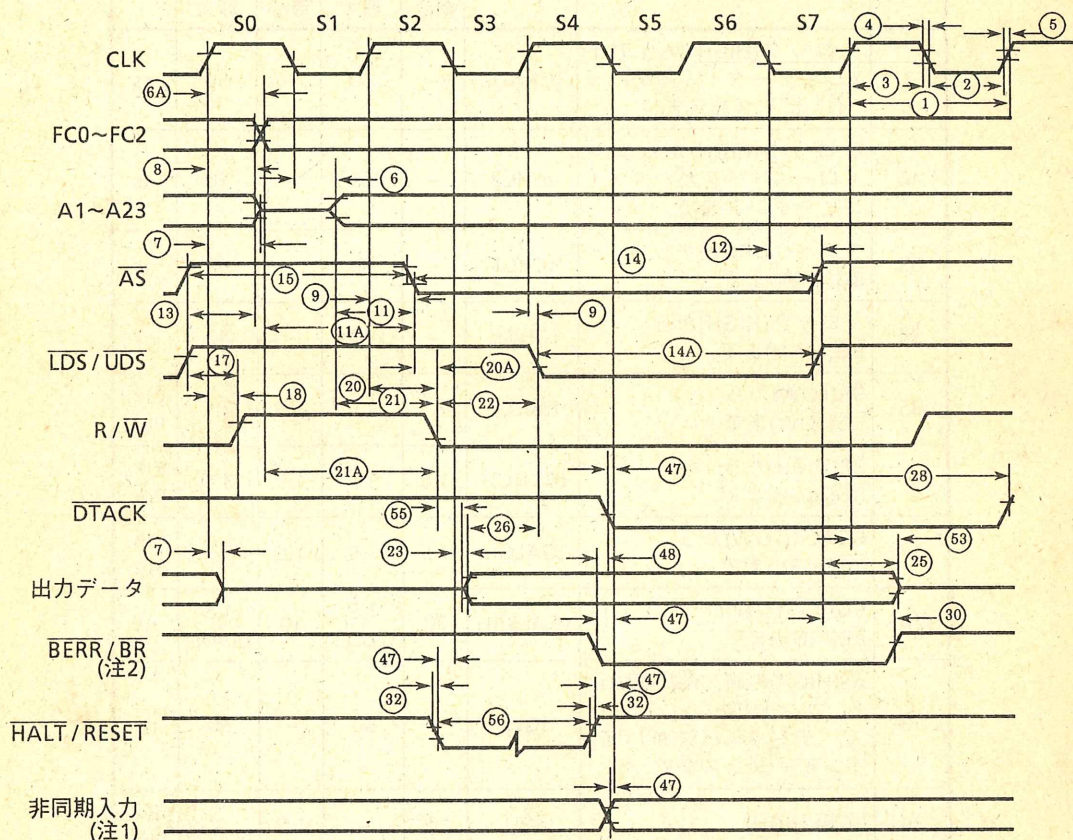
番号	項　目	記号	12.5MHz		16.67MHz		単位
			最小	最大	最小	最大	
46	$\overline{BGACK}$幅(LOW)	tGAL	1.5	–	1.5	–	Clk. Per.
47[5]	非同期入力セットアップ時間	tASI	10	–	10	–	ns
48[2.3]	$\overline{BERR}$(LOW)から $\overline{DTACK}$(LOW)まで	tBELDAL	20	–	10	–	ns
53	クロック(HIGH)から出力データ無効	tCHDOI	0	–	0	–	ns
55	R/$\overline{W}$(LOW)からデータ・バス・ドライブまで	tRLDBD	10	–	0	–	ns
56[4]	$\overline{HALT}$ / $\overline{RESET}$パルス幅	tHRPW	10	–	10	–	Clk. Per.
57	$\overline{BGACK}$(HIGH)から$\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ドライブまで	tGASD	1.5	–	1.5	–	Clk. Per.
58[7]	$\overline{BR}$(HIGH)から制御バス・ドライブ	tRHSD	1.5	–	1.5	–	Clk. Per.

1. 静電容量が50[pF]以下の場合は、Max値から5[ns]を引きます。
2. 実際の値はクロック周期によって決まります。
3. #47が$\overline{DTACK}$と$\overline{BERR}$の両方を満足した場合、#48は0[ns]でも構いません。
4. 電源投入時は、MPUの回路を安定させるために、100[ms]間$\overline{RESET}$状態を保って下さい。電源投入後の最短$\overline{RESET}$状態は、#56を参照して下さい。
5. 非同期セットアップ時間(#47)を満足している場合、$\overline{DTACK}$が低レベルになってからのデータ・セットアップ時間(#31)は無視できます。ただし、データは次のクロック周期に対して、データ入力からクロック低レベルのセットアップ時間(#27)を満足する必要があります。
6. $\overline{AS}$とR/$\overline{W}$の負荷が±20%の間で等しい場合、tASRVの最大値はそれぞれ10[ns] (8～12.5MHz)または5[ns](16.67MHz)引いた値になります。
7. 外部裁停回路が$\overline{BGACK}$をアサートする前に$\overline{BR}$をネゲートした場合、TMP68301は$\overline{BG}$をネゲートし、再びバスを占有します。
8. 最小値において正常な動作が保証されます。最大値を越えた場合は、$\overline{BG}$を再びアサートする可能性があります。

図8.2の波形は、タイミング規定の測定でエッジ間の関係だけに注目しています。入出力信号の機能を説明したものではありません。別の章の機能説明やデバイス・オペレーションのための図を参照して下さい。

(注)

1. 非同期入力$\overline{\text{BGACK}}$, はクロックの立下りエッジで検出されます。
2. $\overline{\text{BR}}$はこのバス・サイクルの終りで認識される場合のみ、このタイミングでアサートする必要があります。
3. 特に指定がない場合、高レベル電圧は2.0[V], 低レベル電圧は0.8[V]としてタイミングを測定してあります。電圧波形は、この範囲の外側から始まり、この範囲内で直線的に変化しなければなりません。

図8.2　読出しサイクル・タイミング

図8.3の波形は、タイミング規定の測定でエッジ間の関係だけに注目しています。入出力信号の機能を説明したものではありません。別の章の機能説明やデバイス・オペレーションのための図を参照して下さい。

(注)

1. 特に指定がない場合、高レベル電圧は2.0 [V], 低レベル電圧は0.8 [V]としてタイミングを測定してあります。電波波形は、この範囲の外側から始まり、この範囲内で直線的に変化しなければなりません。

2. R / $\overline{\text{W}}$と$\overline{\text{AS}}$はS2の立ち上がりエッジでアサートしますが、負荷によってはR / $\overline{\text{W}}$と$\overline{\text{AS}}$より後になることがあります。(#20A)。

図8.3　書込みサイクル・タイミング

8.5 AC特性－バス裁停

(V_{CC} = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; 図8.4)

番号	項目	記号	12.5MHz		16.67MHz		単位
			最小	最大	最小	最大	
7	クロック(HIGH)からアドレス, データ・バスがハイ・インピーダンスまで	tCHADZ	–	60	–	50	ns
16	クロック(HIGH)からコントロール・バスがハイ・インピーダンスまで	tCHCZ	–	60	–	50	ns
33	クロック(HIGH)から $\overline{BG}$(LOW)まで	tCHGL	–	40	–	40	ns
34	クロック(HIGH)から $\overline{BG}$(HIGH)まで	tCHGH	–	40	–	40	ns
35	$\overline{BR}$(LOW)から $\overline{BG}$(LOW)まで	tBRLGL	1.5	3.5	1.5	3.5	Clk. Per.
36[1]	$\overline{BR}$(HIGH)から $\overline{BG}$(HIGH)まで	tBKHGH	1.5	3.5	1.5	3.5	Clk. Per.
37	$\overline{BGACK}$(LOW)から $\overline{BG}$(HIGH)まで	tGALGH	1.5	3.5	1.5	3.5	Clk. Per.
37A[2]	$\overline{BGACK}$(LOW)から $\overline{BR}$(HIGH)まで	tGALBRH	20	1.5 Clocks	10	1.5 Clocks	ns
38	$\overline{AS}$(HIGH)の時, $\overline{BG}$(LOW)から, コントロール, アドレス, データ・バスがハイ・インピーダンスまで	tGLZ	–	60	–	50	ns
39	$\overline{BG}$幅(HIGH)	tGH	1.5	–	1.5	–	Clk. Per.
46	$\overline{BGACK}$幅(LOW)	tGAL	1.5	–	1.5	–	Clk. Per.
47	非同期入力セットアップ時間	tASI	10	–	5	–	ns
57	$\overline{BGACK}$(HIGH)からコントロール・バス・ドライブまで	tGABD	1.5	–	1.5	–	Clk. Per.
58[1]	$\overline{BG}$(HIGH)からコントロール・バス・ドライブまで	tGHBD	1.5	–	1.5	–	Clk. Per.

1. 外部裁停回路が$\overline{BGACK}$をアサートする前に$\overline{BR}$をネゲートした場合、プロセッサは$\overline{BG}$をネゲートし再びバスを専有します。
2. 最小値において正常な動作が保証されます。最大値を越えた場合は、$\overline{BG}$を再びアサートする可能性があります。

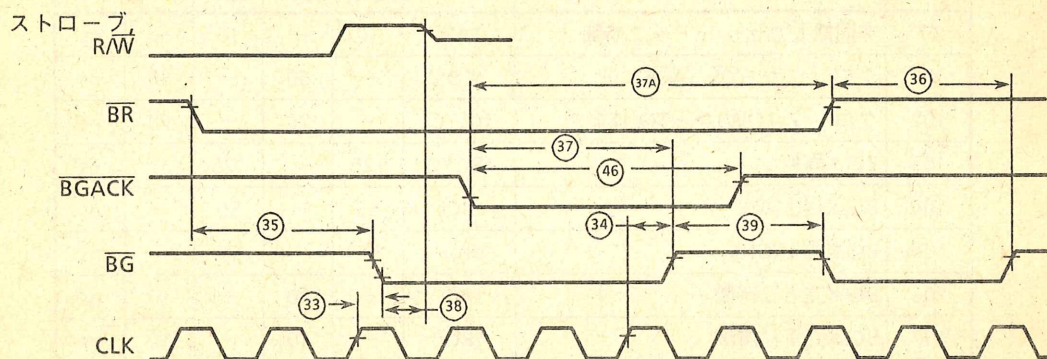
図8.4の波形は、タイミング規定の測定でエッジ間の関係だけに注目しています。入出力信号の機能を説明したものではありません。別の章の機能説明やデバイス・オペレーションのための図を参照して下さい。

(注) 非同期入力BERR, BGACK, BR, DTACKはクロックの立下りエッジで検出されます。

図8.4 バス裁停タイミング

外部バス·マスタが内蔵デバイスのレジスタをアクセスする場合は、読み出し/書込みサイクルのタイミングに従って、アドレス,データ,コントロール信号を入力して下さい。

8.6 AC特性－周辺

(V_{CC} = 5.0V ± 5%, GND = 0V, Ta = 0～70℃; 図8.5～図8.13参照)

番号	項目	記号	12.5MHz		16.67MHz		単位
			最小	最大	最小	最大	
47	非同期入力セットアップ時間	tASI	10	—	10	—	ns
101	クロックから$\overline{CS}$, $\overline{IACK}$まで	tCDS	—	50	—	50	ns
102	クロック(LOW)からTOUTまで	tCLTO	—	70	—	70	ns
103	BCLK周期	tBCYC	125	—	125	—	ns
104	BCLK幅(LOW)	tBCL	55	—	55	—	ns
105	BCLK幅(HIGH)	tBCH	55	—	55	—	ns
106	BCLK立上り時間	tBCr	—	10	—	10	ns
107	BCLK立下り時間	tBCf	—	10	—	10	ns
108	$\overline{LDS}$(HIGH)から$\overline{DTR}$, $\overline{RTS}$まで	tDSMC	—	140	—	140	ns
109	$\overline{DSR}$から$\overline{LDS}$(LOW)まで	tMCDS	50	—	50	—	ns
110	$\overline{DS}$(HIGH)からIO出力まで	tDSIO	—	60	—	60	ns
111	IO入力時のCLK(LOW)に対するIOデータのセットアップ時間	tIOsCL	25	—	25	—	ns
112	IO入力時のCLK(LOW)に対するIOデータのホールド時間	tIOhCL	25	—	25	—	ns
113	DATAのおくれ時間	tDDA	—	60	—	60	ns
114	CLK(HIGH)から$\overline{DSTB}$まで	tCHST	—	60	—	60	ns
115	PRIMEのおくれ時間	tDPR	—	60	—	60	ns
116	$\overline{DSTB}$に対するDATAのセットアップ時間	tDAsST	20	—	20	—	ns
117	$\overline{DSTB}$に対するDATAのホールド時間	tDAhST	60	—	60	—	ns
118	$\overline{DSTB}$幅(LOW)	tSTL	70	—	70	—	ns
119	BUSYおくれ時間	tDBY	—	100	—	100	ns
120	FAULTおくれ時間	tDFA	—	100	—	100	ns
121	CLK(HIGH)から$\overline{ACK}$まで	tCHAC	—	60	—	60	ns
122	$\overline{ACK}$(LOW)からBUSY(LOW)まで	tACLBY	—	60	—	60	ns
123	$\overline{ACK}$(HIGH)からBUSY(LOW)まで	tACHBY	—	60	—	60	ns

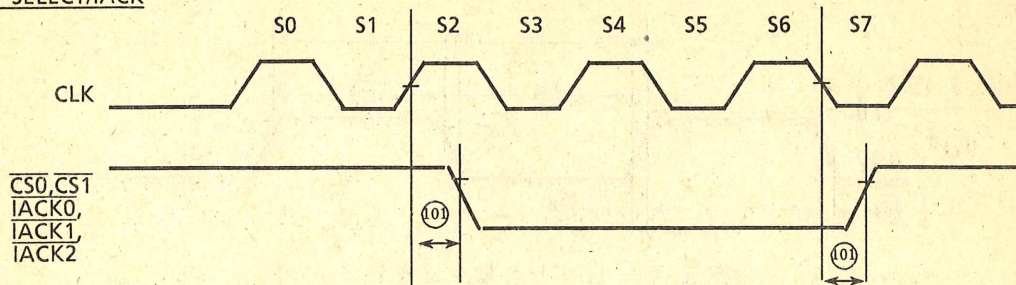



図8.5 CS, IACKタイミング

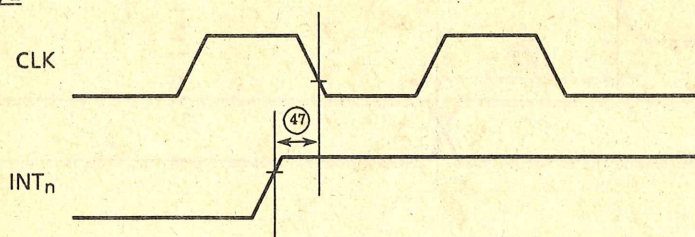



図8.6 割込み要求タイミング

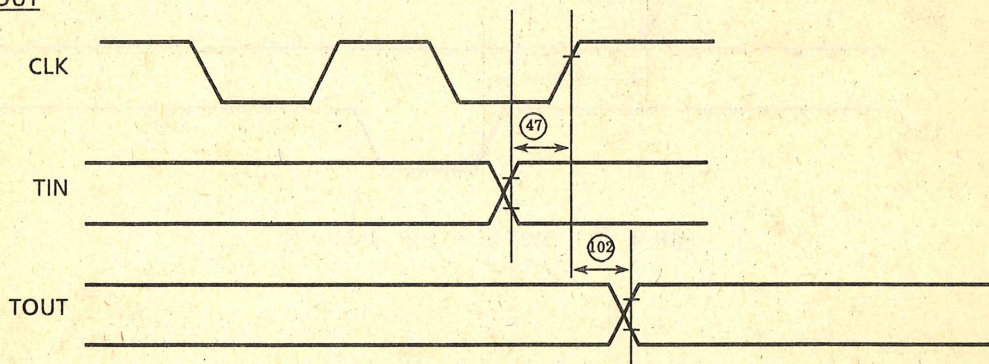



図8.7 タイマ入出力タイミング

BCLK

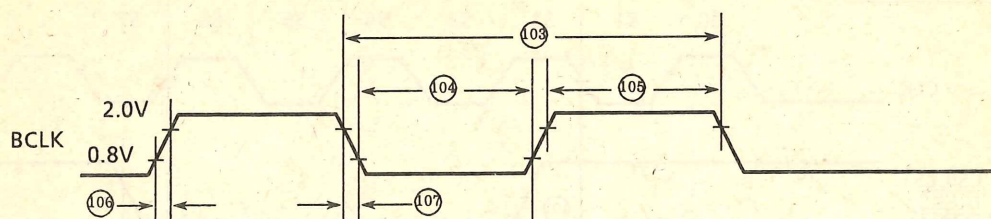
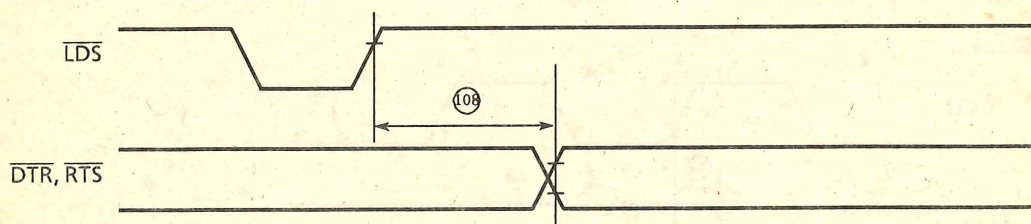



図8.8 ボーレートクロックタイミング

Serial Port Write

Serial Port Read

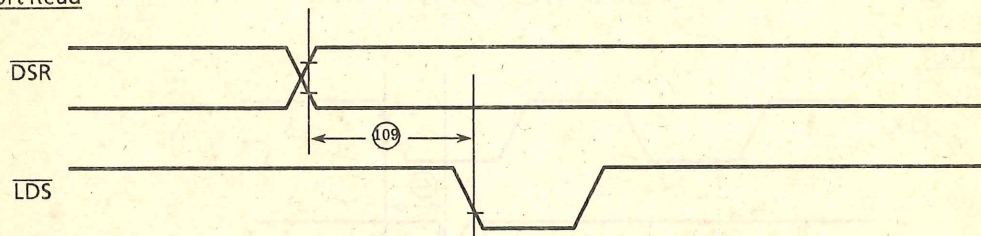



図8.9 シリアルポートタイミング

Parallel Port

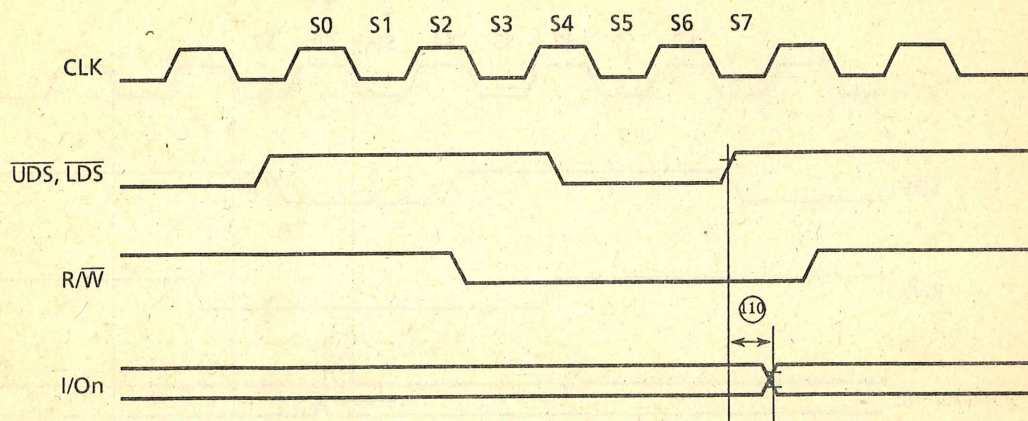



図8.10 IOポート出力タイミング

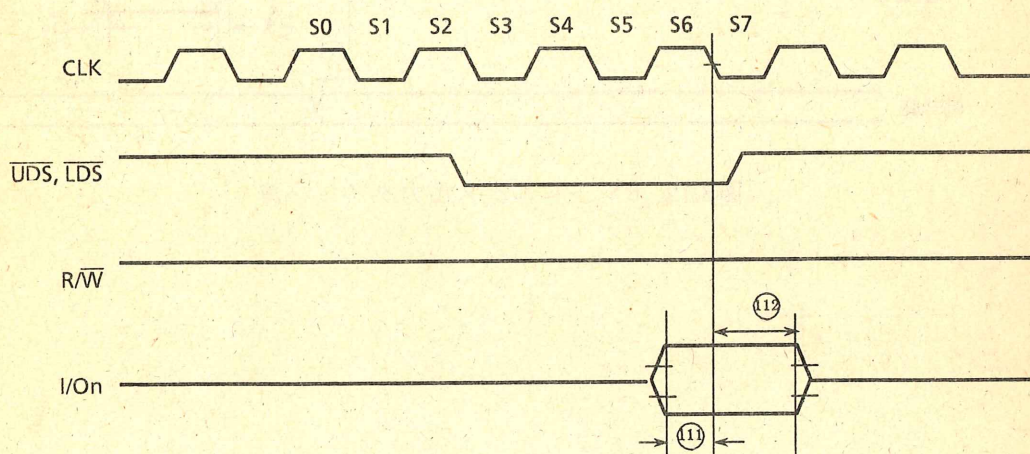



図8.11 IOポート入力タイミング

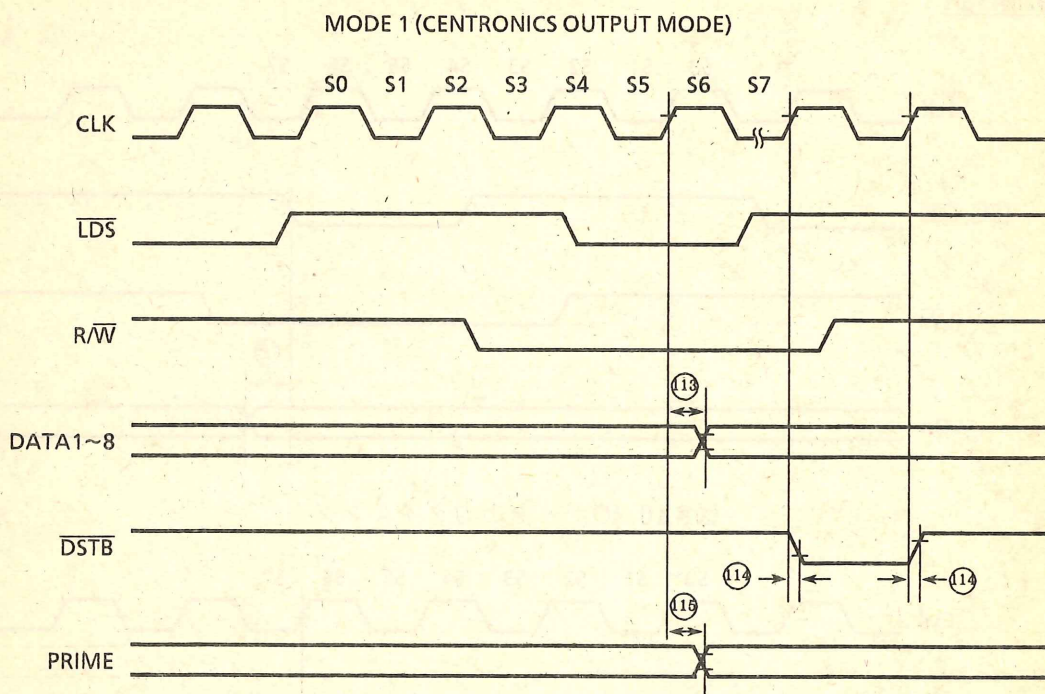



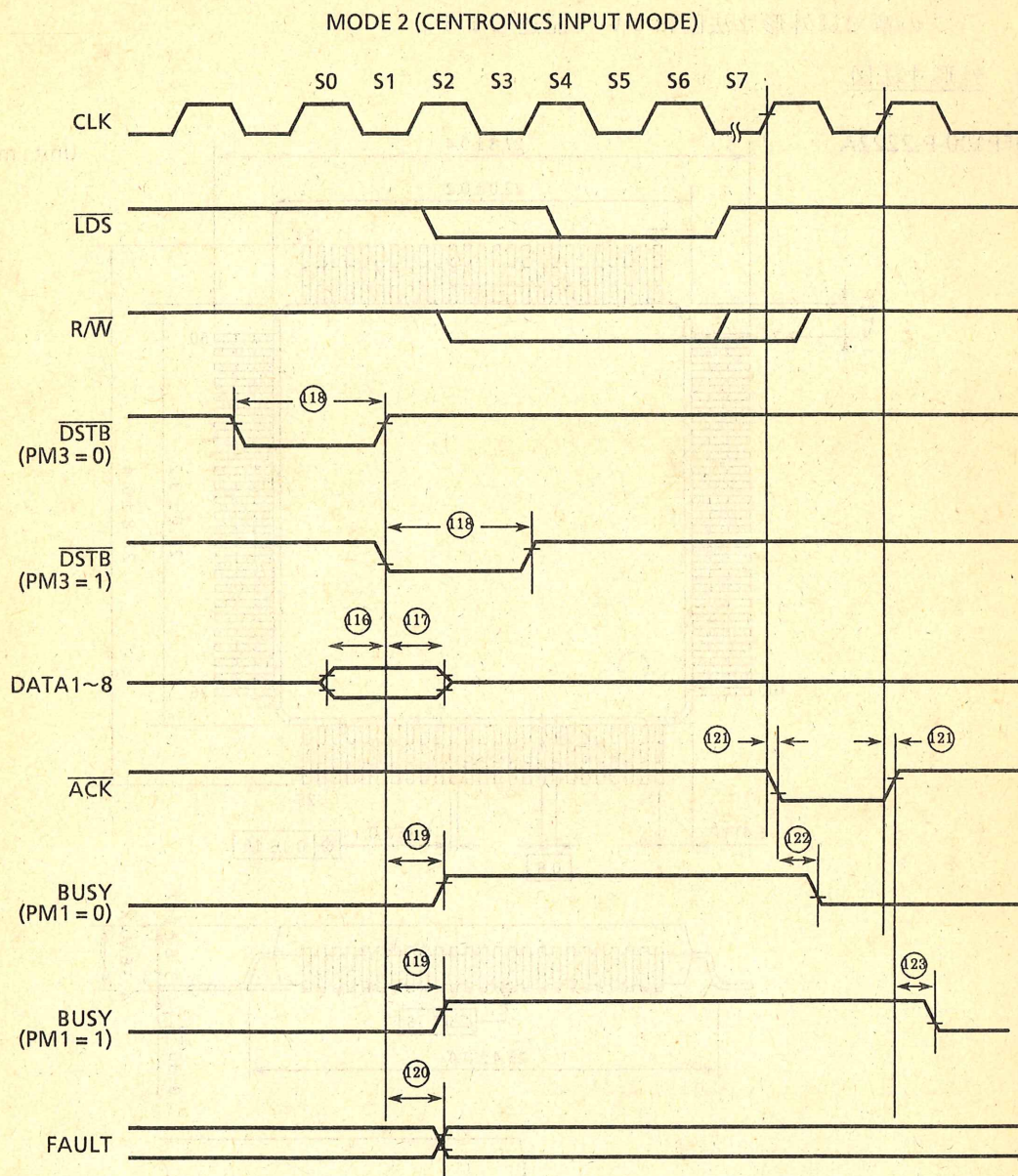
図8.12 セントロニクス出力タイミング

図8.13 セントロニクス・入力タイミング

第9章 機械的データ

この章では外形寸法図について記述します。

9.1 外形寸法図

QFP100-P-2222A

本書に記載されています使用例は説明のために参考として示されているもので、この例を使用することにより生じた諸問題については、弊社およびモトローラ社では一切責任を負いません。

本資料に掲載されている製品は外国為替及び外国貿易管理法に定める戦略物資に該当するため、輸出する場合、同法に基づく輸出許可が必要です。また、本資料に掲載されている製品には米国輸出管理規制の規制を受けた製品が含まれており、輸出する場合、輸出先によっては米国政府の許可が必要です。

記載された仕様およびデータは予告なしに変更される場合があります。